PRINCIPLES & PRACTICES OF ONTOLOGY REUSE

by

Megan Katsumi

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

© Copyright 2016 by Megan Katsumi
Abstract
Principles & Practices of Ontology Reuse

Megan Katsumi
Doctor of Philosophy
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2016

Reusability has always been a defining characteristic of ontologies; it is the key to their supposed benefit of shareability, and the answer to the difficulty of their development. Consequently, reuse is a crucial aspect in the paradigm of ontologies. The vision for reuse is a state in which design via reuse occurs whenever it is possible. In such a state, no unnecessary design work is ever performed, and the supposed benefits of ontologies will naturally result.

Unfortunately, the vision for reuse is not feasible at the time of this writing. Reuse is currently a difficult, somewhat ad-hoc process with unpredictable results, therefore it is often not a viable alternative to designing an ontology from-scratch. This thesis addresses these issues by reducing the barriers and increasing the motivation for reuse. These objectives are achieved by way of 3 key contributions: (1) rigorous techniques for reuse, (2) a formal definition of reuse, and (3) a comprehensive architecture for reuse. Rigorous techniques are designed and presented in such a way as to minimize the demands on the developer, thereby reducing the barriers resulting from the difficulty of the task. The formal definition of reuse is applied to make the benefits of shareability and reduced design work more tractable for the developer, thereby increasing the motivation to design via reuse when appropriate. Finally, the comprehensive architecture describes the infrastructure necessary to apply the collection of solutions presented here in order to simplify their adoption. Beyond this, we consider ways in which the solutions presented here might be extended as the practice of ontology reuse evolves. The work contained here provides a solid foundation upon which the vision for reuse may be achieved and sustained.
Acknowledgements

So many individuals have contributed to the completion of this dissertation in so many ways. As it turns out, this has been one of the more challenging sections to compose.

First, I must thank my parents and my sister as their support extends long before this endeavour. They have been a constant source of patience, understanding, support, and – perhaps most importantly – humour in my life.

Thank you to Royce for constantly encouraging me to pursue my research, along with so many other things in life. Also thank you to Pauline and Wah Lun for treating me like part of the family and for welcoming me, and my mess of papers and books, into their home.

Thank you to my colleagues in the Semantic Technologies Lab, both past and present, for the advice, moral support, and good humour that they have always been ready to provide.

I would also like to thank my committee members, Mark Fox and Li Shu, for continuing to provide their time and support over the last several years. Their questions and thoughts have challenged and inspired me, and helped to shape the final outcome of this dissertation. Further, I am grateful to both Deborah Stacey and Sheila McIlraith who took the time to review this work and participate in my final evaluation. I truly enjoyed the discussion that followed, and their insights will undoubtedly have an impact on the direction of my future research.

Finally, this thesis would not have been possible without the continued support of Michael Grüninger. Michael has always been open to my ideas, and patient with my questions. Our weekly meetings, which we never got around to recording, were always something to look forward to. They were engaging and often helped to motivate me when work became difficult. Enrolling in his fourth year decision support systems course has turned out to be truly life-changing, as this lead to my introduction to the subject of ontologies and so many other wonderfully interesting areas of study. I could not have asked for a better supervisor and am truly grateful for all of the guidance that he has provided me with, in the completion of this dissertation and beyond.
# Contents

## Introduction

1.1 Ontologies ................................................................. 1
   1.1.1 Why Ontologies? .................................................. 2
   1.1.2 Ontology Development .......................................... 3
   1.1.3 Ontology Design via Reuse ..................................... 3
1.2 The Problem with Reuse ............................................... 5
   1.2.1 Significance ....................................................... 5
1.3 Objective ............................................................... 5
   1.3.1 The Problem with Horses and Water ......................... 6
1.4 Approach ....................................................................... 7

## Related Work

2.1 Definitions of Reuse ..................................................... 10
2.2 Rigorous Techniques ..................................................... 10
   2.2.1 Ontolingua – What Happened? ................................. 13
2.3 Comprehensive Architecture .......................................... 14
2.4 Summary ..................................................................... 14

## Reducing the Workload

3.1 Design via Reuse .......................................................... 16
   3.1.1 Search Intuitions ..................................................... 17
   3.1.2 Choice Intuitions .................................................... 18
   3.1.3 Reuse Intuitions ...................................................... 18
   3.1.4 Pragmatic Challenges in the Current State .................... 18
3.2 The Optimal Division of Labour ....................................... 21
   3.2.1 Search ................................................................. 21
   3.2.2 Choice ................................................................. 24
3.3 Summary ..................................................................... 29

## Candidate Preference Based on Semantic Requirements

4.1 Related Work .............................................................. 31
4.2 The Idea ................................................................. 32
   4.2.1 Preference Defined ................................................ 33
4.3 Viability ..................................................................... 36
### 4.3 Assumptions

- 4.3.1 Assumptions
- 4.3.2 The Procedures
- 4.3.3 An Example: Choosing Between Participation Ontologies

### 4.4 Correctness and Completeness of Criteria

### 4.5 Discussion

- 4.5.1 The Sufficiency of CQs
- 4.5.2 H_R Root Theory Definition
- 4.5.3 Implementation Concerns

### 4.6 Summary

### 5 Attaining the Benefits of Reuse

- 5.1 Reusability Defined
- 5.2 Definition of Reuse
  - 5.2.1 Other Approaches to Reuse
- 5.3 Reduction of Design Work
- 5.4 Interoperability
  - 5.4.1 Specialized Reuse Operations
  - 5.4.2 Identifying Necessary Specialized Operations for Reuse
  - 5.4.3 On the Uniqueness of Necessary Operations
- 5.5 Discussion
  - 5.5.1 Current Issues with Reuse
  - 5.5.2 Classification and DOL
- 5.6 Summary

### 6 Examining the Benefits of Reuse in Practice

- 6.1 An Ontology of Dates
  - 6.1.1 Dates in Practice
  - 6.1.2 Using the Date Ontology
  - 6.1.3 Other Approaches to Dates
  - 6.1.4 Dates via Reuse
  - 6.1.5 Summary
- 6.2 A Home Services Ontology
  - 6.2.1 Modifications to gist
  - 6.2.2 Summary
- 6.3 Factors in the Benefits of Reuse

### 7 An Infrastructure for Reuse

- 7.1 Scope
- 7.2 Conformance Requirements
  - 7.2.1 Search
  - 7.2.2 Choice
- 7.3 Supplementary Recommendations
- 7.4 Integration with Existing Solutions for Ontology Development
C  Motivation for Reuse with Translations

C.1  Signature Translation: Identifying Reuse of Ontology Patterns

   C.1.1  COLORE and Ontology Design Patterns

   C.1.2  Relationships between Ontologies in COLORE

   C.1.3  Hierarchies as Design Patterns

   C.1.4  Summary

C.2  Logical Language Translation: Using PSL to Extend Event Ontologies

   C.2.1  Background

   C.2.2  Generic Requirements

   C.2.3  Extensions of the Event Ontologies

   C.2.4  Evaluation

   C.2.5  Summary
Chapter 1

Introduction

Reusability has long been recognized as a key attribute of ontologies, however the actual reuse of existing ontologies remains one of the less developed areas in the ontology community. This is problematic given that many of the motivations for using ontologies rely on the assumption that the practice of reuse is commonplace. It is the goal of this thesis to address the problems with reuse, thereby supporting a future state in which the value of ontological solutions is better motivated.

1.1 Ontologies

Ontologies are artefacts that are developed for knowledge representation; they describe a domain of interest by defining the relevant concepts in the domain, and the relationships between these concepts. Admittedly, in the literature various definitions of the term ontology have been presented and there exists a range of artefacts that may be considered ontologies. These are illustrated in the commonly used ontology spectrum (Figure 1.1). For the

![Figure 1.1: The ontology spectrum, originally developed by [41]](image_url)

purposes of this work the term ontology will refer only to the formal ontologies at the right end of the spectrum,
and the following definition will be sufficient for our use of the term:

**Definition 1.** An ontology is an artefact written in a logical language that formally defines the semantics of a collection of concepts associated with a particular domain of interest, within some universe of discourse.

The “formal” stipulation of this definition is important to note, as these ontologies are formal enough to be transcribed into machine-readable languages, thereby creating the opportunity for a variety of useful applications, such as in decision support systems. With the use of an automated reasoner, an ontology that is applied for decision support answers queries regarding the domain it defines. These queries can not only provide information about the domain in general but also about a particular instances in the domain. For example, given the right information an ontology describing manufacturing processes could draw conclusions about a specific process that is occurring on a particular day. Much of the work in the ontology community focuses on formal ontologies that are essentially constructed as taxonomies. While these artefacts are included in our definition of formal ontologies, it should be noted that our work also includes – and in many cases focuses on – a different, more rigorous sort of ontology; this type of ontology is distinct in that it often does not explicitly define a taxonomy at all. The focus of this more rigorous ontology is on utilising axioms to define, in depth, the intended semantics of its concepts. This axiom-centric, rather than taxonomy-focused paradigm allows the potential to support the execution of more complex inferences, rather than the relatively basic queries supported by the is-a relationship.

### 1.1.1 Why Ontologies?

For many people there is some uncertainty regarding the value of ontologies, and often times this is because there exist other technologies (e.g. databases, computer programs) that can be employed to accomplish the same tasks that an ontology may be applied for. Nevertheless, the value of ontologies is not so much in their ability to solve a specific, individual problem, but rather in the semantics that they provide for a given domain, and the unambiguous, machine-readable way in which this semantics is specified. We identify four major advantages of ontologies over alternative technologies:

1. **Unambiguous:** Ontologies enforce a formal, common and explicit understanding of a domain. The unambiguous specification of semantics makes the application of an ontology a useful exercise in itself, as it forces those applying the ontology to understand and agree on the meaning of all of the concepts in their domain. A well-axiomatized ontology can subsequently detect errors in its application in the form of inconsistencies – if some part of the domain-specific knowledge is inconsistent with the ontology, it can be detected by an automated reasoner.

2. **Adaptable:** Both the content and application of ontologies are easily adaptable for dynamic domains. The semantics of any given concept are constant, therefore in theory the addition or removal of concepts (i.e. to adapt the ontology to a changing domain) is straightforward. The same ontology can be applied to represent data, check consistency of the data and its representation, answer queries, and achieve semantic integration. Alternative technologies on the other hand, such as schemas, database engines, and other task-specific tools are much less flexible in their application. If software is developed to integrate two computer programs, later in its lifetime it can’t easily be re-purposed to store and query domain data.

3. **Reusable:** Reuse is inherent in the notion of ontologies. Owing to the single and explicit understanding of the domains that they specify, the reusability of an ontology in its domain should be expected. This is an intentionally inherent characteristic of ontologies since their original conception [27]. Once the effort
has been put forward to clearly and correctly define some concept, the definition should be reusable for all subsequent applications.

4. **Shareable:** An additional result of the explicit formalism is that it enables the seamless sharing of information across any applications in which it is employed. When an ontology from one application is reused in another, the defined semantics are shared between the two applications thereby supporting semantic interoperability between them. Further, their explicit axiomatizations may also be leveraged to identify and define partial or complete shareability between seemingly distinct theories.

### 1.1.2 Ontology Development

Before delving further into the ways in which existing ontologies are reused in the development process, we should first ensure a clear understanding of the nature of ontology development and implementation, in general.

In many ways, the development of an ontology is not so different from that of a kitchen cabinet. Both are formalized and developed from some related and underlying concepts. Ontologies often require the use and integration of multiple foundational concepts, such as space and time, whereas the kitchen cabinet’s design involves the application of various theories and bodies of knowledge, such as materials science and woodworking. While it would certainly be feasible to apply these principles in such a way as to arrive at a design for some formation of a cabinet, without a particular application there would be no way to say whether or not it is a good or suitable cabinet. The same holds true for the development of an ontology.

In both cases we can make some restricted assessments of correctness or quality based on the design (axiomatization), with respect to the underlying concepts alone. For example, does the cabinet violate any laws of physics? Or, is the ontology’s axiomatization consistent with our understanding of the domain? However, to fully assess either artefact requires consideration of its implementation. The implementation of a cabinet design includes the steps of its manufacture through to its installation and eventual use; similarly, the ontology’s implementation includes the installation of a particular instance of its axioms into the designed architecture, and eventual use. In both cases, it is this eventual use that motivates the requirements that must be defined early on in development. The intended application and resulting requirements provide a constant source of guidance for the design process and eventually, a more detailed criteria against which we can evaluate the resulting design.

While it is true that ontologies are sometimes developed without a specific application in mind – this approach is generally flawed as there is no way to ascertain whether or not the resulting theory\(^1\) is “good” or “correct”. Similarly, it makes little sense to develop a cabinet for an unspecified application. We can do little more than assess whether the design meets the basic criteria of a cabinet. Is it “good”? Is it the right size? The right cost? Will it stand up to the necessary load? All of these assessments require knowledge of the intended implementation. While perhaps less apparent when considering high-level concepts such as time and space, this application-oriented position is valid here too. Even these foundational concept-focused ontologies must be intended for eventual use in some applications – otherwise they amount to solely philosophical exercises, in sharp contrast to the discipline of ontology engineering that motivates much of today’s work.

### 1.1.3 Ontology Design via Reuse

Reuse sometimes refers specifically to the idea that the same ontology is implemented in multiple applications, as in the advantage of *adaptability* described earlier. In this scenario, the requirements in multiple instances are

---

\(^1\)By an ontology, we mean a set of first-order sentences closed under logical entailment. We, therefore, use the words ‘ontology’ and ‘theory’ interchangeably.
met by the same ontology, so that after its initial design the ontology required for the subsequent applications is created by simply reusing the original, as-is. While this may seem idealistic, recall that ontologies were originally conceived as as shareable, reusable knowledge bases \cite{cite}: they were intended to function like building blocks. As described in the previous section, it is this kind of capability that allows ontologies to distinguish themselves from other technological solutions.

The term reuse is often employed more generally to describe the actions taken when a set of previously-developed axioms is somehow incorporated into a new theory. It may also be used to describe the overall process that is involved in the act of incorporating these existing axioms, including for example, choosing which theory is best suited to the needs of the development project. For clarity, we will distinguish between these two concepts by referring to either reuse of some ontology, in other words the specific task, or the process of design via reuse.

Design via reuse is an accepted idea, but not a particularly well-defined concept; it may refer to any number of ways in which an existing ontology is somehow leveraged for the purposes of developing a new ontology. Let us consider again the analogy to the development of a cabinet to illustrate some of the different ways in which reuse can occur.

The same cabinet design may be reused in multiple different instances (e.g. construction projects) to manufacture and install precisely the same type of cabinet. This corresponds to the ideal sort of reuse; when the entire ontology is essentially re-implemented in a different application.

Beyond this, the design of the cabinet may be subjected to more complex kinds of reuse, requiring more expert knowledge such as from a skilled carpenter or furniture designer (the ontologist). The same original cabinet design may be reworked for different applications – each of these may require the addition of different external pieces (other types of cabinets, counters, etcetera), different colours, finishes, or fixtures. In some such cases, the defining qualities of the cabinet (such as its size, capacity, and function) will remain unchanged, while in others the modifications may have a minor to substantial impact on the cabinet’s qualities.

When beginning the design of a new cabinet, or other product altogether, a designer may reuse certain components belonging to the design of the cabinet, such as the design of a hinge or opening mechanism. This could either be due to convenience, or with the aim of taking advantage of some aspect that has desirable qualities such as cost or performance. The isolation and reuse of the component may be straightforward or it may be a more involved process, depending on whether a design specification for the specific component already exists. Similarly, there are instances where only part of an ontology (i.e. a selection of axioms), well-defined or otherwise, is reused for some new design.

More creative instances of reuse can also occur. In some situations resourceful individuals will completely re-purpose the design (with or without modifications) for a completely novel and unforeseen application, such as in the popular “Ikea Hacks” communities. Previous work discussed in Chapter \cite{cite} identifies analogous opportunities for ontologies; there are cases where an ontology from a seemingly unrelated domain may, with a different perspective, provide precisely the axioms required for a particular design project.

Each of these scenarios corresponds to different ways in which ontology reuse might occur. The precise approach taken may be influenced by a number of factors such as the required design, the available designs for reuse, the experience of the designer, and so-on. It is straightforward to see that outcome, in terms of both the resulting work required and the relationship between the new and reused ontologies, will vary substantially across these different approaches.

\footnote{http://www.ikeahackers.net/}
1.2 The Problem with Reuse

Reuse has long been and remains acknowledged as an important means of ontology design [82, 65]. As noted in the previous section, many of the concepts required for a given application are generic in nature, thus one of the major benefits of ontologies is that they should be reusable across different applications. Unfortunately in practice ontology design via reuse is not the norm. We conjecture that this absence of reuse is because it is not yet a feasible alternative to from-scratch development. This view is supported by the review presented in [78], where several case studies are reviewed in depth and issues for reuse are identified.

1.2.1 Significance

The current lack of design through reuse presents a serious problem to the ontology community. As noted, many ontologists are faced with skepticism regarding the value or usefulness of ontologies. While there are a variety of other approaches to knowledge representation, as described in Section 1.1 the case for ontologies is that they provide a knowledge representation that is not only unambiguous but adaptable, shareable, and reusable. Given that the reuse of ontologies is not a common development practice, the benefits of shareability and reusability that we attributed to them earlier are essentially discounted.

It is a given that there will be concepts that appear in multiple applications and thus ontologies are certainly reusable in theory, however this is irrelevant if they are not being reused in practice. If reuse is not commonplace we cannot claim that reusability is a benefit of ontologies. The value of ontologies was also motivated by the benefit of shareability. In general, this assumes that ontologies are developed with reuse. If reuse does not occur, then we cannot expect there to be instances in which information can readily be shared between applications.

Without the benefits of reusability and shareability, we are left with only the benefits of adaptability and an unambiguous representation. While adaptability is still certainly a benefit, it will also be irrelevant in certain scenarios, and an unambiguous representation alone is insufficient to motivate the use of ontologies over other comparable solutions. Without a well-rounded set of benefits, the case for ontological solutions, and consequently the importance of the ontological research community as a whole, is less than motivating.

This absence of reuse indicates that it is either not a desired benefit, or there are factors that inhibit its realization. Assuming that the creators of existing ontologies have been rationally motivated, the former makes little sense and thus we can only conclude the latter. Given that reuse is such a serious problem for the community, it is unclear why it remains unaddressed. Various attempts to aid and facilitate the reuse of ontologies have been made (we review these in greater detail in the subsequent chapter), however whatever their impact has been, they have clearly been unable to solve the problem of reuse. This is by no means an indication of ineffective or poor quality work, but an indication of the difficulty of the problem. The scope of the problem of reuse is broad and poorly defined. In part, this is because reuse itself is currently not well-understood. There exist many approaches to reuse, formalized to various degrees, and yet we have no clear definition of what it means to develop an ontology via reuse or precisely what this entails.

1.3 Objective

The vision for reuse is a state in which ontologies are only designed ‘from scratch’ when no opportunities for development by reuse exist. In other words, reuse will occur as part of the design of ontologies whenever it is possible. Only in the case where no existing work may be leveraged will the developer create an new set of axioms, unrelated (at least in the formal, logical sense) to previous work. In such a state, as the body of publicly
available ontologies grows and evolves, design by reuse will become the norm rather than the exception in the practice of ontology development. We claim that unfortunately, at the time of this writing this vision is not feasible because reuse is currently not a viable alternative to designing an ontology from scratch.

There are many barriers to the task of reuse, and its importance for the community has been insufficient motivation to encourage large-scale uptake. In order to perform reuse, the developer must undertake some additional tasks (i.e., to find and choose the ontology they will reuse). In theory the additional effort required to perform these tasks should be offset by the design work that will be saved, and by the additional benefits that can be expected of the resulting ontology. However, the reality is that these additional tasks are not straightforward, and through a lack of direction may become considerable barriers. Further, if these tasks are not performed correctly we cannot even guarantee that much or any design work will be saved. Ontology design may be a challenging task, however it is simple enough to understand, whereas ontology reuse is not well-defined, nor is the process involved well-understood. Determining what tasks to perform and how is a non-trivial challenge because while there are guidelines for reuse, there are no rigorous techniques, and no comprehensive architectures that define precisely how all of the tasks may be performed and how they fit together. The guidelines that do exist are often incomplete in breadth, and lack sufficient detail to be reliably reproduced. If reuse is to become a viable alternative, we need to ensure that the process and its required tasks are more straightforward so as not to outweigh the potential benefits.

In the previous section we discussed the importance of reuse to realize the advantages of the ontological solution. While this provides motivation for the community, the realization of these benefits is also key to providing immediate motivation for developers to reuse ontologies. However, resulting from both a lack of guidance and lack of a formal definition of reuse, the approaches taken vary widely and in the current state these benefits are not guaranteed. Depending on the approach taken for the reuse of an ontology, we may be able to only partially guarantee some benefits, or perhaps none at all. Developers may be deterred from the process of reuse because of experiences in which the advertised benefits were misleading, however this is not a fault of reuse as much of an inappropriate application of it. To avoid such scenarios, reuse and the resulting operations must be sufficiently defined such that we can precisely identify what is required and predict what the outcome will be.

In summary, currently reuse is a difficult, almost ad-hoc process with unpredictable results. Rather than the benefits outweighing the barriers, we find that in the current state the reverse is true; in many cases there may be only barriers and no benefits whatsoever! To achieve a future state where the vision of reuse is feasible, where reuse is not only a viable but a competitive alternative to design from scratch, the current state must be remedied. First, the barriers must be reduced – the process of design via reuse needs to become easier; it must become straightforward to search for, choose, and reuse existing ontologies. A challenge to achieve this is that there are currently no rigorous techniques for reuse to employ, and for those methods that do exist, there is no comprehensive architecture to describe how they should be implemented in concert to support the entire process. Second, the motivation must be increased – the touted benefits of reuse must be tractable, in other words they must result predictably from its undertaking. This is challenging because it requires a clear and precise understanding of not only what constitutes reuse, but how it is actually performed. This two-part solution to the viability of reuse forms the objectives of the thesis.

1.3.1 The Problem with Horses and Water

Given that reuse is widely recognized as an important issue in the community, we can be optimistic that our solutions will be adopted as there exists substantial motivation for the ontology community to facilitate development via reuse. We claim that by improving the situation – simplifying the process and providing stronger
incentives to perform it, the frequency of development via reuse can be expected to increase. Ideally, we would like to claim that if our solutions are adopted, then reuse will occur whenever it is possible. In other words, that our contribution will be sufficient to achieve the vision of reuse. However, the problem with this claim is that it ignores the wild-card of human nature. Our identified contributions are sufficient to achieve an environment in which the vision of reuse is not only feasible but encouraged, yet we do not and cannot guarantee the behaviour of members of the ontology community.

1.4 Approach

In the previous section, the first objective we identified was to reduce the barriers to reuse, making design via reuse easier to accomplish. For the process of reuse to become easier, the developer’s workload must be reduced. This can be achieved by providing explicit guidance and removing ambiguity from the required tasks. Such explicit guidance requires the identification and provision of rigorous techniques for reuse. Any of these tasks that do not require developer input may be offloaded to some independent service (human or automated), thereby even further reducing the developer’s workload. In order for such solutions to be effective as a whole, some form of a comprehensive guide or architecture to coordinate the solutions and the required tasks must also be prescribed.

The second objective we identified was to increase the motivation to reuse ontologies. To accomplish this we focus on ensuring that the benefits are delivered, predictably, when developing an ontology via reuse. This requires consideration of two perspectives of the problem: both the promises that are made as well as their fulfilment. We develop a precise, formal definition of what operations constitute ontology reuse, and a means of determining precisely what operations are required to reuse a given theory. This allows the developer to make reliable predictions regarding the outcomes of reuse, thus enabling them to recognize when there is sufficient motivation to reuse a particular theory.

In achieving the objectives of the thesis, we will have made the following major research contributions:

1. A formal definition of reuse.
2. Rigorous techniques for reuse.
3. A comprehensive architecture to support the process of reuse.

Following a review of relevant literature in Chapter 2, we begin by tackling the first objective in Chapter 3. In this chapter we present the optimal division of labour to minimize the developer’s workload. We describe how the offloaded tasks may be accomplished by an external resource, through the implementation of rigorous techniques; in Chapter 4 we describe the solution for the particularly challenging task of assessing evaluation results. While the first objective is attainable with only general intuitions of what it means to reuse an ontology, to address the second objective requires a more precise identification of what constitutes ontology reuse. In Chapter 5 we provide a precise definition of reuse and reusability, and then describe how these results may be employed to attain predictable benefits of reuse. Following this, in Chapter 6 we present two examples that illustrate how the different types of reuse impact the potential benefits. In Chapter 7 we define a standard for an infrastructure to integrate and support all of the solutions in practice. This is important as it provides pragmatic guidance for the community on the adoption of these solutions; it serves as a specification of the comprehensive architecture required to implement the solutions to both objectives. It is guaranteed that any infrastructure conforming to these requirements will achieve our objectives. We conclude by considering a broader definition of reuse in Chapter 8. Should other active research efforts be successful, in a future state we may have effortless interoperability across
logical languages. In this final part of the thesis, we expand the definition of reuse to account for reuse with heterogeneous signatures and logical languages and consider how this would impact the solutions presented here. These observations serve as motivation for future work on improving the state of ontology reuse.
Chapter 2

Related Work

Often the question is raised as to why the ontology community does not simply re-purpose techniques for software reuse. First, it is important to clarify that the task of software reuse is still an active area of research, therefore while the tools and techniques may be more mature there are still no complete solutions to leverage. Further, there exists no rigorous definition of reuse in the software community of the sort we require to achieve our objectives. Nevertheless, many applicable design and reuse solutions have already been leveraged: development methodologies, ontology libraries, and even import functionality. However, while there are certainly strong parallels between software reuse and ontology reuse, there is a key difference between the functionality of software and ontologies: ontologies’ functionality is based on reasoning. Even if the ontology itself is not applied for reasoning, the semantics it defines for a concept(s) - the possible interpretations and the possible theorems that can be inferred based on this semantics are all based on reasoning. This is fundamentally different from the functionality of software, therefore we cannot expect that techniques from the software community will be applicable here. Functionality plays a major role in reuse; it impacts the relevance as well as preference between theories, therefore the processes and the challenges of reuse will be substantially different.

It is for this reason that we – along with the authors of much of the related work on ontology reuse – have found it necessary to consider reuse from an ontology-specific perspective. In the previous chapter, we described how despite existing efforts the problem of reuse remains unresolved. The challenges of reuse are broad and complex in nature; it’s likely that in most instances rather than some major flaw preventing existing work from resolving the issue of reuse, the majority of work has failed to solve the problem of reuse simply because its scope does not cover the full breadth of the reuse process and its challenges. Existing work ranges in scope: while some attempt to address the entire process, others focus on a specific task or function. In this chapter, we review this related work and for each contribution we consider whether it addresses any of the specific challenges for the objectives that were identified in the previous chapter; in other words, can any existing work be leveraged to achieve the objectives?

The review presented in [78] is similar in spirit to this work, although it does not provide any solutions for reuse, it attempts to derive a set of requirements for reuse to be a viable design alternative. However we find that the identified requirements, while relatively specific, are incomplete. This is likely a result of the fact that the requirements are derived from a series of case studies which, although thorough, do not comprise a particularly solid foundation. We therefore opted not to use these results to guide the work presented here, (nonetheless, it would be interesting to assess how our solution compares to the author’s proposed requirements). We take a different perspective in our review of existing work: having already identified the objectives for reuse at a high
level, we consider the challenges that must be addressed to achieve them and assess whether any existing efforts are applicable.

2.1 Definitions of Reuse

As noted in the introduction, there is currently no generally accepted, concrete definition of ontology reuse. It appears that nearly all work pertaining to the task of reuse assumes some implicit definition of the reuse of an ontology; no effort is made, formal or otherwise, to provide further clarification of precisely what this entails. An exception to this is found in the sort of definition presented by [7], where the authors describe reuse as:

...the process in which available (ontological) knowledge is used as input to generate new ontologies.

This is a very broad definition, and its lack of specificity not only limits its usefulness but, we will see later, is also inaccurate in its generality. A similar, implicit definition may be found when reviewing the guidelines for reuse in [20], where the authors’ customizing activity (Activity 2) accounts for a wide range of loosely defined operations (pruning, enriching, translating, and adapting the selected ontology). Without any justification or precise definition of what these operations entail, it appears as though the authors also consider reuse to include any scenario in which an available ontology is used as an input in the development of a new one. Attempting to extract an implicit definition from other existing guidelines only reinforces that this vague definition is generally accepted by the community. At best, we find some more specific, related, and sometimes overlapping subtypes of reuse have been defined, such as merging and alignment [61], integration [22, 23], and modular or ‘safe’ reuse [26]. Even with the provision of examples and guidelines, these implicit definitions remain either vague or isolated to a specific type of reuse. No existing work has provided a concrete understanding of precisely what is, and what is not ontology reuse.

2.2 Rigorous Techniques

We have claimed that a lack of rigorous techniques presents a substantial challenge for simplifying the task of reuse. In this section we consider what sorts of rigorous techniques, both theoretical (i.e. methodology-oriented) and implemented (i.e. tool-based), are currently available.

In [23], the authors not only provide a definition of ontology integration, but prescribe a methodology to perform it. Termed the ONIONS methodology, the focus is on resolving term matches; case-by-case, the authors prescribe an appropriate resolution depending on if or how such terms are related. While logical, the scope of the method is limited; even within the task of integration the assumption appears to be that the content from both ontologies should be contained in the resulting, integrated theory. This precludes instances where, for example two ontologies have conflicting ontological commitments and thus one or both terms should be eliminated. An alternate perspective is taken in [26]. Here, the authors focus on cases where term matches result in a concern of unintended consequences (i.e. the ontology being developed affecting the meaning of concepts in the reused ontology). The authors define a notion of ‘safe reuse’ based on the idea of a conservative extension. This concept is defined with respect to some part of the ontology’s signature; it is a way to guarantee that the ontology will not affect the semantics of some set of terms, should an ontology that defines them be imported. This concept is rigorous and well-defined; the authors provide sufficient but incomplete conditions that may be implemented to check for safety in practice. While safety is certainly a desirable property to have, it is not likely that all cases of reuse will conform to this definition. The provision of this criteria (though restricted to OWL) serves as a
useful tool for developers should they wish to design an ontology with safe reuse, however it does not address the challenges of reuse in general.

In [70, 71, 72] the authors develop a methodology for ontology reuse; specifically, a restricted class of reuse identified as “integration”. The authors prescribe a selection of candidates based on a set of strict and desirable features. After the candidates are identified they are assessed from two perspectives: the domain experts’ technical evaluation of the knowledge represented, and the users’ (ontologists’) assessment of ontology-specific properties such as structure, quality of definitions and documentation. Taking these analyses into account, the final choice of ontology(s) to reuse is made based on the independent suitability of the ontology (its content and other qualities), and its compatibility and completeness with respect to the desired end result. The approach seems both practical and reasonable; in fact, the activities identified and described provide the most complete consideration of reuse available in current literature. While this work may serve as an effective high-level guide for reuse, its situation at the “knowledge-level” (abstraction of the implementation level) limits its potential beyond this. The methodology and operations identified are consequently high-level and not formalized to a degree such that they may be employed consistently in practice. This resulting approach, though useful, lacks rigour; more detail is required at the implementation level to provide adequate support for reuse.

In [20] the authors utilise a template to specify the development requirements, describing how it should be used to search for potential candidates. Specific attention is paid to the use of functions in the Watson search engine [10], however apart from this the guidance is relatively general. Although a structured template could certainly be useful, there exist such a wide variety of sources that the general guidance provided here is of limited use in practice. Interestingly, we speculate that this may be the reason that other guidelines tend not to focus much attention on the task of search. Much of the existing work neglects the task of search altogether, essentially presuming that the candidates are a given in the problem of reuse. In practice we find this to be an oversimplification; this task of candidate search possesses its own set of challenges that are discussed further in Chapter 3. However, it may be that this omission is not an oversight but a practical decision in order to avoid committing to the use of a specific source of candidates. This contribution also includes the task of choosing between candidate ontologies, (Tasks 1.3 and 1.4 in the context of their work). The most appropriate ontology for reuse is eventually determined through the evaluation of a weighted calculation of what the authors refer to as functional and non-functional features. Although such a calculation appears objective, the features’ values are assigned based on subjective assessment (e.g. code clarity: unknown, low, medium, or high). This approach is not well-founded and it is too informal to provide accurate, reliable results.

In [8], the authors propose to support reuse with the provision of what they term ‘contextual information’ – this is essentially ontology metadata that the authors argue will serve to aid in identifying whether an ontology is appropriate for reuse. While this may be true, its restricted scope does not cover the key issues of reuse, and even still pragmatic issues relating to its implementation were not resolved.

CORE, an environment to support ontology reuse using a repository is proposed in [19]. Here, suitable ontologies for reuse are recommended based on lexical and taxonomic similarity measures that are calculated between the ontologies in a repository and a set of terms that is provided to represent the domain of the required ontology. The tool returns a ranked list of candidates, essentially addressing both the task of search and choice at once. Although it neglects the consideration of other types of requirements, such an approach might be useful to support the task of search as its functionality is beyond what is offered by traditional keyword search engines. On the other hand, some improvements to search engines have been made and suggested in existing work; we discuss these later in this section. CORE also attempts to provide techniques to support the choice between candidate ontologies. Though we observe the tool has potential to support the task of search, its ranking criteria
would likely be of limited use to support the task of choice in practice. Not only are the criteria used relatively superficial, but since the tool does not account for the range of requirement types typical of a development project the resulting assessment would be incomplete. Similarly, the OntoQA tool [80] attempts to calculate useful metrics from the features of an ontology in order to inform the decision regarding its suitability. Yet another metric-based technique is presented in a methodology called OntoMetric [54] which focuses on the use of a comparison framework as a means of supporting the task of choice; in other words, this methodology is to be applied after a set of candidate ontologies has already been gathered. The framework presented by the authors consists of set of approximately 160 characteristics, divided across 5 dimensions (content, language, methodology, tool, and cost). The user then specifies their requirements by determining how to extend or prune the given framework. A process is then described through which the framework may be used to calculate a weighted value to aid in comparing and selecting between ontologies. A corresponding tool was also implemented to support this methodology. As with CORE, we find that the criteria employed for these techniques are too superficial to provide an accurate assessment of the appropriateness of a set of candidates.

Upper Ontologies  One tool to support ontology reuse comes in the form of upper ontologies, such as DOLCE [22], the Suggested Upper Merged Ontology (SUMO) [60], and OpenCyc [55]. Axiomatizations of “foundational” concepts that are common to most domains, such as time and space, are specified with the intent that they be reused to provide the foundational axioms for virtually any domain ontology. Upper ontologies have the potential to be reused for a range of applications - in particular to provide axiomatizations of more general concepts. However, there are some drawbacks to this approach. There exist a variety of upper ontologies that define the same types of foundational concepts, but in different logical languages, and with different ontological commitments (e.g. using different theories of time) thus often times it may be difficult to distinguish which ontology is the most suitable. These ontologies are also typically quite large, therefore if good documentation is not provided it is difficult to achieve a sufficient understanding of their semantics in order to make an informed selection. Upper ontologies have failed to provide an effective means for facilitating widespread reuse; in part this is a result of issues inherent in their reuse paradigm [38], but also due to the fact that the provision of axioms alone is insufficient to support the entire process.

Repositories  An alternative to using an upper ontology to provide axiomatizations of the required concepts is to select ontologies from a repository. A review of these tools was provided in [11], highlighting the range and variance and identifying useful features to compare and distinguish between them. Although many of the efforts reviewed by this work are no longer active, the general observations made by the authors remain accurate. Similar to upper ontologies, repositories also typically contain foundational or application-independent concepts, however in a repository the axioms are organized into smaller, focused ontologies (sometimes modules). Since the ontologies within a repository do not need to be consistent with one-another, a repository can contain ontologies with different ontological commitments. Different repositories have varying degrees of utility for reuse, as there are many with different purposes, metadata, and organization of ontologies. Although it is no longer active, one of the first repositories was created in the form of the Ontolingua server [17]. We discuss aspects of the greater Ontolingua effort in the following section.

A more current repository is COLORE [35] [36], in which the ontologies are organized into modules, and model-theoretic relationships between theories in the repository are provided where available. This information could be helpful in selecting potential ontologies to be reused for some development effort. The Linked Open
Vocabularies (LOV) effort takes a similar approach for lightweight ontologies. Owing to the limited expressiveness of the representation language, the relationships defined between theories differ, however the intuition is the same.

There are also domain-specific repositories, such as Bioportal which provides access to biomedical ontologies via browse, search, and visualization functionalities. It also facilitates community involvement for the evaluation of these ontologies, such as through user reviews. While these characteristics provide useful information for the identification and selection of ontologies for reuse, neither Bioportal nor any of the other repositories available enforce a straightforward and effective search or selection process.

In general, repositories are a promising means of supporting the identification and comparison of ontologies for reuse, however the functionality provided in the current state is insufficient to completely address these tasks.

**Search Engines** Search engines, such as Watson and Swoogle are another approach to providing tool support for the identification of reuse candidates. Some attempts have been made to assist in the search of ontologies through an informed ranking of the search engine results based on the links between them. Examples of this may be found in Swoogle and earlier work by ; both attempts are analogous to Google’s PageRank algorithm. These techniques are interesting and useful improvements on alternatives such as basic term matching in a repository, or the use of a traditional web search engine to find ontologies. However, they are an oversimplification of the ontology developer’s requirements and thus fail to address the more complex challenges of identifying and choosing between appropriate ontologies for reuse. In general, search engines are distinct from repositories in that they crawl the web, as opposed to searching their own internal library; there are advantages and disadvantages to each approach which we will not discuss here. For our purposes, these engines perform nearly the same function as repositories, with varying search abilities; neither fully supports the rigorous techniques required for reuse.

**Design Patterns** Inspired from software engineering, ontology design patterns (OPs) are concerned with identifying recurring aspects of ontology implementation and offering design solutions in the form of different types of patterns. These patterns may be reused directly, or as a template to generate content for the ontology being developed. In six families of OPs are presented: Structural OPs, Correspondence OPs, Content OPs (CPs), Reasoning OPs, Presentation OPs, and Lexico-Syntactic OPs. Existing work focuses primarily on CPs, as illustrated by the content of the Ontology Design Patterns website which is currently the primary OP catalogue available. Patterns in the catalogue are organized by pattern type and name, accompanied by a set of metadata that varies with the OP type. This metadata, while potentially useful, is not complete for each entry. Further, when compared to repositories we find the information and relationships between the various OPs provided to be relatively informal. This approach may be a less practical tool for reuse if there already exist ontologies that are suitable for describing some larger portion of the domain of interest; since each CP is relatively small, multiple CPs would need to be selected and integrated instead of selecting and integrating only one ontology.

2.2.1 Ontolingua – What Happened?

Ontolingua was the first notable attempt at an ontology repository, but its contribution extended to a much broader scope. Ontolingua constituted an entire ontology development environment – one among many such tools that were available at the time. It consisted of its own language, an extension of the Knowledge Interchange Format
(KIF) language [23], which was intended to support portability between representation languages, and a set of tools for ontology development. While the heterogeneity of logical languages still poses an issue today, it was a much greater challenge in the time of Ontolingua as there were no true de facto standard representations for the various logics. In addition to storing ontologies in this language, Ontolingua provided users with a set of tools, most notable for reuse were those for assembling and extending ontologies. In essence, Ontolingua was meant to be an environment for development via reuse. A review of this early work finds many key solutions and suggestions for challenges of importing theories and disambiguation between terms, which are reflected in current practice.

Although the contributions of Ontolingua are insufficient to address the challenges identified here – its solutions do not address the entire process of reuse, nor does it present a formal definition of reuse – we have seen in this review that no subsequent approaches have fully addressed these issues either; in other words, it did not fail because it was superceded by a better solution. While its role as a repository falls short due to a lack of relationships between ontologies, the Ontolingua effort made many key contributions, so it is surprising that it did not continue. We speculate that the failure of Ontolingua was not a result of any short-comings in its contributions, but a result of changes in the landscape of the ontology community. Specifically, when OWL gained traction and eventually became the standard for the Semantic Web, the Ontolingua language was de-valued as there was less need for the existence of multiple logical languages and formalisms. At the same time Proteg´e gained popularity as a development environment, likely owing to its user-friendly interface, integrated reasoning support, and eventual implementation of an OWL plug-in. Ontolingua on the other hand, was a design-oriented environment and did not facilitate the same ease of interaction and direct reasoning with ontology projects. While Ontolingua was not immediately irrelevant with these changes in the community, it is likely that these changes and its failure to match the usability of Proteg´e eventually derailed the momentum of Ontolingua. This serves as a poignant example of the importance of accounting for user goals and user behaviour: regardless of the theoretical and technical quality of a solution, if the barriers outweigh the perceived benefits, the chances of adoption will be low.

2.3 Comprehensive Architecture

Several guidelines exist that attempt to prescribe how various tools and techniques should be used collaboratively in order to perform reuse. Here, we are interested in work that spans the entire process; in other words: what contributions exist that aim to guide the implementation of solutions for design via reuse.

In [72], the authors introduce techniques for the identification and selection of candidate ontologies, however this guidance is provided at a relatively high level, without consideration of other aspects of a complete architecture such as the source of the candidates.

A guideline aimed at supporting the entire process for the reuse of generic ontologies is presented in [20]. While this guideline constitutes a relatively complete architecture, its prescribed techniques (discussed in the previous section) are insufficient to address the challenges of reuse, therefore it is doubtful whether its generalized architecture will be sufficient for the solutions to our objectives.

2.4 Summary

This review has confirmed that existing work fails to satisfy the key challenges of reuse. We find no common, explicit definition of the task of reuse in existing work. We speculate this may be a key cause of the subsequent
scope issues identified in existing techniques and guidelines. How can we expect solutions to cover a scope which essentially does not exist?

Many of the rigorous methods and techniques are flawed or lack sufficient detail, in other words they are either incorrect, or not rigorous. A major shortcoming of the current state is the final assessment and comparison between candidates: the choice between candidates is most commonly addressed via guidelines or tools that are either relatively informal, or metric-based. Given the variety of possible reuse scenarios and the necessary subjectivity in the assessment and decisions to be made, the provision of relatively general guidelines may seem justifiable; however, this position is short-sighted. The complexity that is faced when attempting to make a well-informed decision with such wide variation is so great that guidelines lacking explicit detail are not able to provide much benefit. Although metric-based methods enable a more definite set of instructions, they provide little assurance of accuracy given the subjective nature of both the requirements and their assessment.

While some contributions are promising - in particular those techniques for the storage and identification of relevant ontologies, any potential impact is limited without a comprehensive architecture in which they may be implemented. Currently, architectures for reuse solutions are found only in the form of the guidelines in which techniques are presented. While this in itself is not an issue, we found the scope of all existing guidelines and/or their techniques to be lacking in some areas, thus there are currently no comprehensive architectures for the entire task of development via reuse. Again, this issue is not surprising given the lack of a definition of reuse.
Chapter 3

Reducing the Workload

Earlier, we claimed that in order for reuse to be a feasible design alternative and thus potentially achieve the vision of reuse, the workload of design via reuse needs to be reduced for the developer. It is straightforward to see that making design via reuse easier will encourage its uptake (or conversely, discourage its avoidance). What may be less clear is how this can be achieved. We begin by clarifying what is meant by design via reuse and then motivate and explain precisely how we intend to make this process easier for the developer. Then, we introduce a straightforward criteria to define the minimum work required and present a solution for the optimal division of labour that results.

3.1 Design via Reuse

First, the process of design via reuse requires more detailed consideration. Here, we consider how reuse occurs in the context of traditional ontology development. In any sort of structured development process we have at minimum some form of (1) Requirements Specification, (2) Design, and (3) Evaluation. Reuse of an ontology is essentially a special case of the Design; after the requirements have been specified, the developer may select an existing ontology to reuse in the design of the axioms, rather than create them completely from scratch. This option is often acknowledged in the literature, although little attention is paid to the details of the process itself [62, 18]. Unless, a priori there is a single target ontology for reuse, it is straightforward to see that several additional steps are required to facilitate this special case of Design:

- **Search:** Some form of search (whether with a tool, or through a mental catalogue of known ontologies) is required in order to identify which ontology(s) might be suitable candidates for reuse.

- **Choice:** There may be several potentially reusable ontologies (reuse candidates) that the developer will need to choose between.

- **Reuse:** Once an ontology is chosen it may be implemented directly in a sort of plug-and-play fashion, but more likely its axioms (potentially other aspects as well such as documentation) will be subject to some additional design work in order to achieve an ontology that satisfies the requirements.

We can recognize the steps of Search and Choice as elements that are unique to design via reuse. These tasks occur in addition to those of traditional ontology development. The Reuse of an ontology is in fact simply an instance of the task of ontology Design. So from a procedural standpoint it is the tasks of Search and Choice that
distinguish ontology reuse from traditional development. Once the candidate ontology is chosen, the procedure essentially returns to ontology development as usual; we are assessing whether the set of axioms we have designed (reused) satisfies the requirements, if not then some modifications must be made and the results will again be assessed until we are satisfied with the final design.

Figure 3.1: Design via reuse in the context of traditional ontology development.

Note that, as illustrated in Figure 3.1, these tasks are completely contained within the Design phase of development. The task of Requirements Specification is no different whether ontology development is occurring via reuse or from-scratch. In fact, there is no reason that the requirements should differ either; the desired content and characteristics of an ontology remain the same regardless of its source. The only notable difference might be observed in the inclusion of some additional requirements that are only applicable to already-existing ontologies. For example, requirements of a candidate’s maturity or popularity might be motivated by considerations of quality or potential support for semantic integration; for obvious reasons these sorts of requirements are only applicable in cases of reuse. In general, the requirements considered in the process of reuse are the same as those previously defined in the literature: content-oriented requirements (sometimes referred to as functional, content-specific, or semantic) and non-functional requirements (NFRs) [18, 49, 79].

There are various levels and types of approaches to specifying requirements for an ontology’s content, however for the purpose of reducing the workload we restrict our focus to the concept of semantic requirements as defined in [49] due to the fact that they are precise, well-founded, and implementable; they also follow from the popular, well-accepted approach of competency questions (CQs) defined in [82]. Semantic requirements may be defined through a characterisation of intended models, or approximated with competency questions. In either case they may be specified with a set of sentences, thus for convenience we denote the set of semantic requirements by $T_R$. All other aspects of the development process are also independent of whether there is reuse or not. With this understanding, we consider the intuitions of each of the tasks comprising design via reuse in order to motivate and provide context for the solution that follows.

### 3.1.1 Search Intuitions

Given a set of requirements, there may be a number of aspects that we would like to incorporate in our search against some repository(s). Independent of existing and proposed approaches, consider the task of search as a completely manual process: we have access to some collection (physical or otherwise) of ontologies. We also
have access to the specification of requirements. We review these requirements and “keep them in mind” while sorting through the collection of ontologies, selecting any that seem as though they might be related. This decision might be based on an ontology’s title, some grouping that it is stored under, or perhaps a closer examination of its documentation or concepts. This basic intuition behind the task of candidate search is also quite similar to that of using search engines on the web: given some criteria (e.g. keywords) the aim is to return a set of relevant results. With each ontology returned by the search process, a conjecture of relevance is being made. For example, we might select a particular ontology for further consideration because we conjecture that some of its axioms could be used to satisfy some of the requirements. Naturally, the goal is not to return just any set of relevant ontologies – the familiar aims of high precision and recall apply here as well.

3.1.2 Choice Intuitions

The task of choice is complex; it requires the consideration and assessment of a variety of objective and subjective requirements – criteria that often have interdependencies as well as individual, implicit and explicit priorities. Independent of any particular methodology, the task of choice can be described as follows: given some set of potentially relevant ontologies, the developer must choose which is the ‘best’ with respect to some explicit and/or implicit requirements. Owing to the aforementioned variety of potential requirements, the precise interpretation of what it means to be the ‘best’ is unclear. Intuitively though, in the context of reuse the ‘best’ choice is the one that requires the least effort in order to satisfy the developer’s requirements.

3.1.3 Reuse Intuitions

Once an ontology (or set of ontologies) has been chosen for reuse, the clear next step is its actual reuse. This task may be conceptualized as re-entering the traditional Design phase with an ontology partway through development. The ontology(s) chosen for reuse is the current version of the ontology being developed, and there is some feedback from the evaluation of requirements that will serve as a guide for required corrections. We make revisions to the chosen ontology(s), as necessary to achieve a version that suits our needs (i.e. satisfies the defined requirements). To satisfy the semantic requirements, we may need to add, remove, or modify certain axioms. For the non-functional requirements, the necessary revisions might include things such as the addition or augmentation of documentation, or translation to an alternate representation language.

3.1.4 Pragmatic Challenges in the Current State

The question of whether the tasks described above are truly an issue in the current state may be raised by those outside of the ontology community. To illustrate the difficulties in a more concrete way, let us consider a scenario in which we wish to reuse an existing ontology to develop a theory of Actors.

While some approaches, such as the CORE search tool presented in [19], assume a single source of ontologies, in practice we are faced with a variety of both means and sources for search. As discussed in Chapter 2 there are a number of both repositories and ontology-specific search engines available to locate existing ontologies; each source often represents a different community, and offers different means of user interaction. We cannot presume to make a well-informed decision by simply selecting one such source. On the other hand, it would not be practical to consider all possible sources of ontologies. Therefore we are forced to compromise, in an effort to be both thorough and pragmatic, we restrict our search to several prominent repositories and search engines.

As discussed in Chapter 2 the search functionality of ontology sources is currently more or less limited to keyword-type search. While some provide alternate means of search customization (for example, whether
it is metadata or code that should be considered in the search), only high-level content requirements can be captured in the search criteria. More precise requirements, such as competency questions, cannot be leveraged by existing sources of ontologies. Using the available search functionality, candidate ontologies are identified from the selected sources for ontologies using the keywords “actor” and “agent”. Although an ontological distinction might be made between these two terms, we opt to include both to guard against omitting any potentially useful theories. Certainly there are other related terms which might also have been used – however again, in the interests of practicality we restrict our search to these two terms as we speculated they would yield the best results. Our search results, organized by source, are summarized as follows:

- **Word-of-mouth:**
  - EEO (Actor Ontology)
  - TOVE (Organisation Ontology)
  - OntoSTIT
  - DOLCE/DnS/DUL

- **Upper ontologies:**
  - SUMO
  - OpenCyc

- **Selection of Open Ontology Repository (OOR) members and other known repositories:**
  - COLORE: no results from superficial search, (it was later found that OntoSTIT is stored here)
  - Ontohub: no results from ontology or symbol keyword search for “actor” or “agent”
  - MMI-ORR:
    * OOI Reference Model
  - BioPortal:
    * RCD
    * SNOMEDCT
    * ROLEO
  - Swoogle:
    * foaf

---

1. [http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html](http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html)
2. [http://ontology.eil.utoronto.ca/organization.html](http://ontology.eil.utoronto.ca/organization.html)
3. [http://colore.oor.net/onto_stit/agency.clif](http://colore.oor.net/onto_stit/agency.clif)
4. [http://www.ontologydesignpatterns.org/ont/dul/DUL.owl](http://www.ontologydesignpatterns.org/ont/dul/DUL.owl)
7. [http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html](http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html)
8. [http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html](http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html)
9. [http://ontology.eil.utoronto.ca/organization.html](http://ontology.eil.utoronto.ca/organization.html)
10. [http://colore.oor.net/onto_stit/agency.clif](http://colore.oor.net/onto_stit/agency.clif)
11. [http://www.ontologydesignpatterns.org/ont/dul/DUL.owl](http://www.ontologydesignpatterns.org/ont/dul/DUL.owl)
14. At the time of writing, some sources were excluded due to technical difficulties, i.e. web pages not operational
16. In BioPortal, we omitted results from the keyword search for “agent” as the number of results was too large to feasibly sift through, and superficially most appeared to be (unsurprisingly) skewed towards the scientific sense of the word.
CHAPTER 3. REDUCING THE WORKLOAD

- vcard-rdf\textsuperscript{14}
- aktors\textsuperscript{15}
- inference web (iw)\textsuperscript{16}
- pim (contact)\textsuperscript{17}
- pml-provenance\textsuperscript{18}
- daml services\textsuperscript{19}

- DAML Library: No longer active, but still a potential source
  - cobra\textsuperscript{20}
  - fipaowl\textsuperscript{21}

Note that the first grouping of search results is not in fact a source of ontologies in the traditional sense, as described in Chapter 2. In a sense, word-of-mouth is a source of ontologies that pre-dates the more current repositories and search engines, however this does not make it any less valid; in particular, it is interesting to note that the ontologies identified through word-of-mouth do not appear in the results of any of the other sources. This points to a general issue of recall in current tools.

The volume and variety of sources alone presents a barrier to reuse; they increase both the workload on the developer who must sift through existing sources, and the chances that a useful ontology may be missed. Further, the limitations of the keyword approach to search mean that additional effort is required to account for alternate search terms (e.g. synonyms or morphemes) such as ‘Agent’. This restricted nature of the keyword search also contributes to low precision of candidates retrieved. Since candidates are found based on a single term match, many irrelevant candidates are likely to be retrieved. This problem becomes particularly apparent when searching for a more generic concept, or one with multiple senses. For example, in the results for Bioportal, the term ‘Agent’ appears with such a high frequency in the ontologies and their documentation (despite the ontology itself being focused in some other domain) that we are forced to exclude the results altogether. This creates an additional burden in that many unnecessary results are retrieved which must then be manually filtered out.

A considerable amount of effort is required to completely review and assess each candidate ontology. Given the large number of candidates retrieved from search, it is impractical to assess every ontology in this manner. A sort of refinement process will be necessary to prune the initial set of results to a more manageable number. We must discern at a relatively superficial level whether there are any candidates that could be excluded from the more detailed evaluation and comparison with respect to the requirements. Unless the candidates all satisfy comparable subsets of the requirements (i.e. there is a clear best ontology), this assessment will not be straightforward. There are functional and non-functional requirements to assess and compare – and admittedly there are likely some implicit requirements that we may have missed in error. With the exception of a case where a single ontology is clearly the best match for the requirements, the choice of which ontology to reuse (or whether to reuse any at all) will require a complex consideration of the tradeoffs that will be made for each choice. To make matters worse, we lack sufficient methodologies to assess the candidates in a consistent, reliable way. Both the evaluations and their assessment are subject to human error; if an error is made, there is an added concern that it may not be
discovered until it is ‘too late’ to correct. For example, we may misjudge the effort involved in correcting some issue. However, too much effort may have been invested, or perhaps the development project will be completed (with more effort than was necessary, or with less than ideal results) before it is ever discovered that an error was made.

In summary, we see that the sources for ontologies and the search criteria they support lead to a necessarily ad-hoc approach to Search. It is at the developer’s discretion which sources to include, and how to most effectively utilize the keyword search. The goal of high recall conflicts with the demands of practicality, and this challenge extends to the task of Choice. We previously recognized that the metric-based approaches to choice are insufficient, yet we find the alternative, thorough assessment and comparison of each candidate to be impractical. In the current state, the ability to both effectively and efficiently design via reuse is out of reach.

3.2 The Optimal Division of Labour

With the goal of making reuse easier for the developer, the optimal division of labour is one in which the developer is required to perform the fewest possible tasks. In the process of reuse, the developer’s involvement is necessary for tasks involving subjective judgements and decision-making. Such tasks may never be completely offloaded to automation or other tool support without risking inaccurate approximations of the developer’s judgement. If all such subjective decisions are isolated, what remains is objective and, we claim, has the potential to be fulfilled by some other resource. The approach taken here is to determine how all of this remaining work can be offloaded to some independent resource, thereby devising an approach to reuse that places the minimum demands on the developer.

In the previous section, we discussed how the process of design via reuse fits into the traditional ontology development paradigm. Two tasks emerged that are unique to Development via Reuse: Search and Choice. Our analysis focuses on these two tasks as they represent the work that is required in addition to the tasks of traditional development, therefore it is these tasks that are the potential source of barriers to reuse. Reuse itself is not included as it corresponds to traditional ontology design. In the following sections, we isolate the subjective tasks and then describe - first for the task of Search and then for Choice - the objective tasks remaining. For each of these objective tasks we identify how it could be supported or performed with some independent resource. The optimal division of labour is implicit in the identification of subjective and objective tasks; the identification of techniques for the objective tasks provides a means of realizing such a division and thereby making the process of reuse as easy as possible for the developer.

3.2.1 Search

Recall that the basic intuition of Search can be defined as follows: given some specification of requirements, the developer is tasked with collecting a set of relevant candidate ontologies. There are two variables of this task that affect how it is performed:

1. the criteria employed in the collection process, and
2. the source(s) being searched

In other words, the two key aspects are where the developer looks and how they look.

The requirements themselves are the only subjective aspect of the task of Search, and as mentioned their specification is outside of the scope of design via reuse. Once specified, the search criteria can be extracted and applied to retrieve a set of candidates in a completely objective manner.
Search Criteria

Any theory that is selected as a candidate ontology by the search criteria essentially corresponds to one or more conjectures of relevance. For example, if I select a particular ontology, $T_A$ from my collection of ontologies, I am doing so because I believe it is relevant, that it ‘matches’ some part of my requirements in some way; I am implicitly conjecturing that $T_A$ is relevant to the aspect(s) of the requirements where I perceived a match.

In the context of ontology development various types of requirements may be specified, however relevance is a requirement on content and therefore for the purposes of search we are concerned only with the content-specific (semantic) requirements. The developer uses these to (manually or otherwise) sort through the collection of ontologies and arrive at a set of conjectures of which ontologies are relevant candidates and how.

If a theory is relevant, we should be able to describe the way in which its content corresponds (i.e. maps) to our requirements. In other words, if a theory is relevant, it must be interpreted by some subtheory of the semantic requirements; if not, then we can conclude that it is irrelevant with respect to the requirements we have specified. We can formally describe a conjecture of relevance as a translation definition(s) from the candidate’s own signature to the signature of $T_R$. These definitions are conjectures that a theory is interpreted by some part(s) of the requirements.

It follows that the criteria for Search may be formalized simply as the signature of $T_R$; a theory is retrieved as a candidate if a mapping(s) to the signature of $T_R$ (recall, this is denoted $\sigma(T_R)$) may be conjectured. Each candidate comprises not only the ontology but the conjecture(s) of its relevance via a set of translation definitions that maps (a subset of) the signature of the candidate to (a subset of) the signature of the requirements. Note that it is in fact possible to derive multiple candidates from the same ontology by making multiple different conjectures of relevance, (i.e. different sets of translation definitions for $T_R$).

In order for an ontology to be a candidate it must have at least one conjectured mapping (translation definition) to the signature of the set of semantic requirements. The current approach to search does this implicitly; when searching through a repository for ontologies with some keyword (presumably an important concept in the ontology), the results are essentially a set of alternative ontologies with a single direct mapping to the keyword.

While the current approach of keyword search is not a particularly challenging task, if the aim is to retrieve relevant ontologies with high precision and recall, (and it is), the task of Search becomes much more cumbersome to do well. In most cases it is not feasible to completely specify an accurate condition for relevance with a single term. Even in a specialized domain, there is the possibility for morphemes and synonyms to come into play. Often times there may be many relevant concepts as the requirements may cover several domains. To perform all necessary searches and aggregate the results requires considerable effort on the part of the developer. A procedure to perform this task would mean that it could be offloaded (potentially automated) in its entirety and thus completely removed from the workload of the developer.

While the task of of developing conjectures may seem inherently subjective, for the ontology developer the process is an objective one. The exact approach to the generating mappings (i.e. conjectures) based on the search criteria is an implementation consideration, not one requiring subjective, case-by-case, decisions from the developer. Given a set of requirements and a criterion for identifying mappings, the task is completely objective and well-suited to a procedural approach that can be offloaded to some external resource.

Not only is such a procedure feasible, but there is a range of possible implementations – varying with respect to how the conjectures are identified. For each alternative, we could specify a different procedure, and for each such procedure we could prove its correctness and completeness (with respect to a given repository), relative to

---

22For a detailed discussion of the meaning of an interpretation, see the definition of a signature translation presented in Chapter.
the specification of the conjecture type. However, as each candidate output by search is simply, by definition, a conjecture of a potentially relevant theory, it makes no sense to discuss the correctness or completeness of the output in the absolute sense. How then, do we know what these alternative means of conjecture specification are, and how can we decide which of these is best? Again, we emphasize that this is an implementation decision. It is a heuristic to be defined that has no bearing on any of the formal definitions or subsequent procedures we specify; it is a decision based on consideration of the tradeoff between exhaustivity and efficiency in the search for candidates.

The following is an example of an implementation with a simple heuristic for generating candidates. This procedure searches only for term matches based on the semantic requirements – in other words it only generates candidates with one-to-one translation definitions between terms in the signature of $T_R$ and terms in the candidate’s signature. A candidate is conjectured only when there is a direct match in both the term name and its arity. This procedure can be formalized as follows:

**Procedure 1** An Example of Candidate Retrieval

**Input:** A source(s) of formally verified (minimum consistent), modular ontologies: $R_1, ..., R_n$

A set of semantic requirements, formalized in a logical language as competency questions: $T_R$

**Output:** A list of conjectures to consider: candidate ontologies $T_1, ..., T_n$ and their corresponding sets of translation definitions $\Delta_{11}, ..., \Delta_{nj}$ to $\sigma(T_R)$

1: for Each collection of ontologies being considered do
2:   for Each ontology in the collection: $T_C$ do
3:     for Each term $\phi_i$ in $T_R$ do
4:       if $\phi_i \in T_C$ then
5:         Conjecture translation definition $\Delta_{C_i} : \phi_i^{T_R} \iff \phi_i^{T_C}$
6:       end if
7:     end for
8:   if $\exists i \Delta_{C_i}$ then
9:     Add $T_C$ and set of translation definitions $\Delta_{C1}, ..., \Delta_{Cn}$ to list of conjectures.
10: end if
11: end for
12: end for

Clearly the example procedure defined above could result in the omission of potentially relevant candidates in some cases – however, there may also be scenarios in development via reuse when the desired signature is known and thus this would be sufficient. A more thorough approach to develop conjectures might look beyond basic matches to include synonyms and morphemes. We could even look at candidates outside of the context of our requirements and conjecture mappings simply based on term arity matches, (this might lead to reuse scenarios similar to the work presented in [39], where the same ontology is reused in a variety of domains).

Translation definitions need not be restricted to one-to-one term mapping. There are many different approaches that might be incorporated to identify possible *sentences* that might be used to define terms in in the candidate’s signature. For example, while a candidate may not possess a relevant concept of *mother* explicitly in its theory, we may make the conjecture that the candidate’s concept of a *woman who has a child* is in fact relevant (equivalent) to the concept of mother. At the most thorough end of the spectrum, we find an approach with a procedure that will run infinitely if some terminating criteria is not met. In this procedure for each term in $T_R$ and for each theory in the repository, we consider all possible sentences in the language of the ontology as translation
definitions for different possible candidate conjectures.

This second approach is clearly not a reasonable one; it illustrates that there is a balance that should be found between exhaustiveness and practicality. Yet we need not sacrifice rigour for efficiency; there are certainly opportunities to implement clever ways of generating good, thorough conjectures. For example, we could implement some type of learning from previous users’ translation definitions; we could leverage relationships between theories to make additional conjectures; we could apply a relatively basic conjecture generation approach with an option for user-override to allow for additional conjectures the user may want to test out. There are countless approaches to conjecture generation that could be implemented in a procedure for search. While there is no single correct procedure, there will certainly be some methods that will be better than others, and perhaps some more appropriate in different contexts. What is important is that some effort is made to design a heuristic that will provide a useful, thorough set of candidates. While the recall of a procedure will be limited by certain feasibility considerations, it is high recall that will contribute to the objective of increased reuse benefits by minimizing the opportunity for the omission of candidates with the potential to facilitate these benefits. Finally, it is important to note that the possible conjecture criteria implementations are also dictated by the sources used. For example, to implement a procedure that retrieves additional conjectures based on mappings between ontologies would require the use of a source(s) that provides this information.

Candidate Sources

In Chapter 2, we provided an overview of the different types of sources for ontologies. While it is certainly encouraging that so many such efforts exist, the number and variety of sources also adds a substantial workload to the task of Search. In the current state it is the developer’s responsibility to select and access the source(s) to be searched, and to ensure a thorough search in which an effort has been made to identify and consider a variety of possible sources. It is straightforward to see that this task is mechanical in nature; it does not require any decision making on the part of the developer and thus could easily be offloaded to some external resource thereby removing the burden from the developer.

3.2.2 Choice

The task of candidate choice is inherently subjective. The designer evaluates the candidates retrieved via search with respect to some set of requirements. We assume that in accordance with some form of development methodology or best practices, the developer will endeavour to specify a detailed and accurate set of requirements that may be used to assess the candidates’ adequacy or ‘fitness’ and thus inform their decision.

While it is fairly certain that there will be some requirements that are not specified, the developer’s decision is necessarily made based on the information available to them. Based on the candidates’ evaluation results, a choice is made as to which, if any, of the candidates is suitable for reuse. Apart from scenarios in which a single candidate is clearly the best alternative, this decision will likely require some consideration of implications and weighing of trade-offs between alternatives. Although this final decision is necessarily subjective, the process preceding it is not. Much of the evaluation of requirements and even some assessments of these evaluation results can and should be made as objective as possible. This is important not only for the goal of minimizing the workload, but also to ensure consistency and reliability of results. We divide the task of Choice into two distinct parts: (1) Requirements Evaluation, and (2) a Cumulative Assessment where the results of the evaluation(s) are considered in order to reach an informed decision.
Requirements Evaluation

As recognized earlier, there are two types of requirements to be considered in a given ontology development project: semantic requirements and non-functional requirements. Analogous to the functional / non-functional divide in software engineering, the semantic and non-functional requirements correspond to the logical and non-logical properties of an ontology, respectively.

Non-functional requirements (NFRs) are a valuable tool in any development scenario as they provide an avenue to specify qualities the ontology should have; they allow for a more complete description of the entire artefact, beyond its semantics. Because of this their evaluation is not necessarily straightforward, sometimes it may require a detailed inspection of both the ontology and its associated documentation – their flexibility means that the evaluation of NFRs is, in general, somewhat ad-hoc. While admittedly a challenging task in traditional ontology development, this ad-hoc evaluation of NFRs becomes nearly infeasible in the case of reuse. For each candidate, it may not be clear exactly how the NFRs apply, and it may also be difficult to evaluate them consistently across all candidates. Apart from this, the sheer volume of work is increased because there are multiple candidates, each requiring this specialized assessment.

Unfortunately, owing to their broad and varied nature, there does not and cannot exist a provably complete, universal set of NFRs for which some standard means of evaluation can be devised. The evaluation of subjective NFRs must remain the responsibility of the developer; further, any objective NFRs for which the required information is not known to be available cannot be offloaded as the evaluation may either be not possible or not assessable in some standard manner. However, assuming the availability of the necessary information, any objective NFRs can be evaluated in a completely objective manner. This ensures accuracy and consistency in the results as the evaluations will be comparable across the candidates. In other words, with respect to a set of assumed metadata, we can identify the maximum objectively assessable NFRs. The evaluation of this set of objective NFRs would be straightforward, and a procedure or some independent party could be employed to test and evaluate the metadata according to each defined NFR. This does not in any way restrict the use of other NFRs for ontology development; the implications are only that all NFRs outside of this set require more effort, and will likely be evaluated with less accuracy. The maximum we can offload to some independent resource are those with guaranteed metadata.

The task of evaluating semantic requirements for from-scratch ontology development has been addressed by previous work [51, 49]. The requirements for a semantically correct ontology may be defined using the relationship between the intended models for the ontology and the actual models of its axiomatization (see Figure 3.2).

**Definition 2.** An axiomatization $T_{onto}$ for a class of structures $\mathcal{M}^{intended}$ is semantically correct if and only if for all models $\mathcal{M}$,

$\mathcal{M} \in Mod(T_{onto})$ iff $\mathcal{M} \in \mathcal{M}^{intended}$.

In other words, an axiomatization is semantically correct if and only if it does not include any unintended models, and it does not omit any intended models. Formally, the potential semantic errors that prevent an ontology from being semantically correct are defined as follows:

**Definition 3.** An error of superfluous models is present in the ontology $T_{onto}$ if and only if there exists a structure $\mathcal{M}$ such that

$\mathcal{M} \in Mod(T_{onto})$ and $\mathcal{M} \notin \mathcal{M}^{intended}$.

$\mathcal{M}^{intended}$ denotes the class of intended models, whereas $Mod(T)$ denotes the class of all models of an ontology $T$. The signature $\sigma(T)$ of an ontology $T$ is the set of all nonlogical symbols that appear in $T$. 

---

23$\mathcal{M}^{intended}$ denotes the class of intended models, whereas $Mod(T)$ denotes the class of all models of an ontology $T$. The signature $\sigma(T)$ of an ontology $T$ is the set of all nonlogical symbols that appear in $T$. 

Figure 3.2: The relationship between intended models for an ontology and the models of the ontology’s axioms (from [32]).
Definition 4. An error of model omission is present in the ontology $T_{onto}$ if and only if there exists a structure $\mathcal{M} \in \mathcal{M}^{intended}$ and $\mathcal{M} \notin \text{Mod}(T_{onto})$.

Competency questions (CQs) are a well-accepted means of requirements specification for ontologies [31, 34, 83]; they represent sentences that the ontology should be able to entail, thus they indirectly impose requirements on both the scope of an ontology’s concepts and its semantics. For example, when specifying the requirements for a time ontology, we might have a CQ such as: (CQ-1) ‘is there some time interval that contains a unique timepoint?’, or (CQ-2) ‘is there some timepoint that cannot be contained by any interval?’ If we were developing an ontology for a more general application, perhaps we might have something like: (CQ-3) ‘is there some process that can only occur at a single timepoint?’. Recent work [51] interprets CQs more specifically as semantic requirements and provides a methodology for their evaluation with the use of an automated theorem prover in the process of ontology development. The evaluation of these entailment problems requires the CQs to be encoded in a chosen logical language and vocabulary, for example CQ-1 might become:

$$(\exists x)(\forall t_1, t_2)\text{interval}(x) \land \text{timepoint}(t_1) \land \text{timepoint}(t_2) \land (t_1 = \text{beginof}(x)) \land (t_2 = \text{endof}(x)) \supset (t_1 = t_2)$$

At the core of this use of CQs is the relationship between the consequent of the entailment problem, and the models of the axiomatization.

The evaluation of any ontologies retrieved by search against these requirements is completely objective and relatively straightforward. The evaluation of semantic requirements amounts to the assessment of a set of entailment problems; developer decision-making is not required to perform this task. When the semantic requirements are implemented for evaluation they take the form of sentences in the consequent of entailment problems. There may be cases where the signature varies from that of the candidate ontology(s), however as discussed in the task of Search, an ontology is selected as a candidate based on a conjecture of relevance that is formally defined as a mapping between signatures. Therefore, in order to have recognized the ontology as a candidate, the conjectured mappings between the signatures must be known and thus may be applied by some independent resource to translate the candidate ontology and facilitate its evaluation. The process is mechanical in nature and could easily be offloaded to some external resource.

As mentioned, the evaluation of these requirements amounts to nothing more than the evaluation of a set of entailment problems. Earlier work has shown that developer participation in the evaluation feedback loop can serve to inform both their Requirements Specification and Design [51, 49], thus it would certainly not be advisable to prohibit the developer from participating or interacting with this process. However with the goal of minimizing the developer’s workload in design via reuse, we observe that the developer’s involvement is not necessary to evaluate the semantic requirements. The evaluation of the sets of entailment problems is objective and thus should be offloaded to an external resource.

Assessment and Final Choice

In order to reach a decision for the task of Choice, the developer must consider the results of the requirements’ evaluation and attempt to draw a conclusion regarding the best option. Is one candidate easier to understand than another? Will it require fewer additions? What about modifications? Is it necessary that the original semantics be preserved? What if one candidate is much better known and accepted in the community? The weighing of these criteria is necessarily subjective and will vary from situation to situation and from person to person; it requires
consideration not only of the result of each requirements’ evaluation, but of the trade-offs and interaction between them. For example, in some cases a great enough degree of notoriety may be a sufficient factor to outweigh added development work; however in other cases the preservation of semantics may be perceived as so important that no degree of notoriety would be sufficient cause to select a candidate with semantics that could not be preserved. Equally possible is the case where preservation semantics may be unaffected by some criteria, but potentially overridden by others. All of these results play a role in the subjective decision of choice – the ontology developer must somehow mentally aggregate all of the assessments for the candidate ontologies and determine which, if any, is suitable for reuse. It is easy to see how this task may become unwieldy, even when comparing only a few candidates.

For example, if we were to set out to reuse an ontology of time for some development project we would encounter a number of very different alternatives. In COLORE alone, there currently exist a total of 67 different theories of time (Endpoints, Date-Time Vocabulary, and OWL-Time for example). All of these alternatives are verified representations of time, but logically they are quite different. Since no single ontology is inherently better than the others, how are we to evaluate their differences to inform our choice? This task can be daunting as it demands a deep understanding of the alternatives and their potential effect on the development required. In many cases, the choice will have a significant impact on the semantics of the resulting ontology as well as its potential for interoperability with other ontologies. The task of assessing and choosing between alternative ontologies is a nontrivial one and presents a major obstacle for developers, should they wish to consider reuse as a means of ontology design; in the face of this challenge it is often preferable to create one’s own ontology rather than to invest the time and effort to understand the alternatives, while also taking the risk of reusing an inappropriate theory.

Clearly, the final choice is the responsibility of the developer. Only they may know precisely the nature of the requirements; in some cases these may even be implicit, incorrect, or incomplete. The developer(s) alone must be responsible for assessing how the tradeoffs between different requirements should be considered; given the inherent subjectivity and complexity of the task, any attempt to approximate this decision with a procedure could not be provably correct or complete. As we saw in Chapter 2, such approximations often amount to no more than superficial assessments; they are presumptuous and risk introducing errors that will hurt, rather than help the state and perception of reuse.

The developer should instead be provided with information to aid their decision; while we cannot approximate their decision, we can provide various objective renderings of comparisons between ontologies that factor into their decision. Due to their informal nature, a meaningful, collective assessment of NFRs is necessarily subjective. Apart from some small gains in the reduction of work through the inclusion of visual aids or aggregates, this task cannot be offloaded from the developer’s workload. Conversely, due to the formal nature of semantic requirements, it is possible to design some objective collective assessments.

One benefit of semantic requirements is that their evaluation is objective and relatively straightforward. However, this standard approach to semantic requirement evaluation is only capable of providing an isolated assessment for each candidate. In order to select between ontologies, the developer must compare the evaluation results of all of the candidates. In the context of CQs this is not straightforward. We cannot infer that a theory $T_1$ is better than $T_2$, simply because it satisfies a higher number CQs than $T_2$; it could in fact be the case that it will be much easier for the developer to extend $T_2$ than $T_1$ to satisfy all of the CQs. To accurately evaluate the CQ results across a set of candidates requires a more thorough analysis to obtain a deeper insight. Such an assessment requires substantial effort on the part of the developer; further it is difficult to ensure its accuracy. Owing to the formal nature of semantic requirements, this assessment of relative effort required can in fact be formally
defined with respect to the logical properties of the ontologies. We define a procedure to obtain provably correct and complete results for such an assessment in the following chapter. With this contribution, we can classify this assessment as a completely objective task and thus the assessment of the CQ results may be offloaded from the developer’s workload. The resulting division of labour for the task of Choice can be seen in Figure 3.3. Note that the division of labour also indicates an evaluation component capable of identifying the reuse operations required and the resulting preservation of semantics. These results will be addressed in detail in Chapter 5.

![Figure 3.3: Choice: Requirements Evaluation and Assessment.](image)

### 3.3 Summary

Based on a subjective-objective divide of the tasks involved in design via reuse, we have identified the optimal division of labour depicted in Figure 3.4. Not only does this approach reduce the workload, in several cases the tasks may be accomplished with greater accuracy and effectiveness when isolated from the subjective tasks of the reuse process. The minimum possible workload for the developer includes only the tasks of the non-standard and subjective NFR evaluation, NFR results assessment, and the final cumulative assessment and decision of which candidate ontology to reuse. All other tasks are completely objective and therefore may be offloaded to some other agent, (potentially human or automated). The solution presented here achieves the objective of making design via reuse easier for the developer, thus greatly reducing the issue of required effort as a barrier to reuse. Further, the techniques presented here improve on the current state of reuse in general, they provide a means of improved accuracy, consistency, and effectiveness with which tasks such as CQ assessment may be performed.
Figure 3.4: A high-level diagram of the developer-infrastructure workload divide.
Chapter 4

Candidate Preference Based on Semantic Requirements

We saw previously that in the process of design via reuse, there is often a choice between multiple candidate ontologies. The developer’s requirements are evaluated and then assessed in order to determine which ontologies are the most preferred choices, given of some set of alternatives. In the previous chapter, we described the difficulty of effectively assessing and interpreting the semantic requirement evaluation results across this set of candidates. Even if we are solely considering the content, a simple aggregation and comparison of the number of requirements is inadequate to capture our intuitions of the most preferred candidate. Can we choose one candidate over another simply because it satisfies more competency questions? What if two candidates satisfy the same number of competency questions, but they correspond to different requirements – should they be assessed as comparable? Clearly more knowledge is required in order to make an accurate judgement; the answer will depend on which requirements are and are not satisfied. Currently, this assessment is the responsibility of the developer, however not only is it prohibitively laborious, it is potentially quite complicated and thus prone to error. We claimed that this task could be offloaded from the developer because, at its core the assessment being made is an objective one. In this chapter we explore the assessment of preference over candidate ontologies in more detail. First, we consider precisely what it means to prefer one candidate over another; we provide a formal definition for relative preference in this sense, and based on this definition we present a provably complete and correct procedure for computing this assessment over a set of candidate ontologies and semantic requirements. The procedure operationalizes the definition of preference, thereby demonstrating the feasibility of offloading this assessment as prescribed in the previous chapter.

4.1 Related Work

The challenge of choosing between candidate ontologies is itself, not novel. While some of the existing approaches to reuse presented in Chapter 2 address the task of choice, no solutions to date have been presented that address the challenge of assessing the candidates’ semantics. Whether implemented as a tool or presented as a guideline, the majority of existing work [19, 20, 54] employs some sort of calculation over a set of subjectively-weighted criteria. This may be a useful indication for some requirements, however these approaches are too superficial to account for the candidates’ semantics. An ontology is not necessarily preferred over another simply because it satisfies more requirements – such an assessment requires a deeper consideration. In contrast, the
approach in [72] gives the ontologies’ content a more thorough consideration. Unfortunately, the approach is labour-intensive, requiring considerable input from subject-matter experts and ontologists, and so it does not address the other half of the issue: the effort required for the comparison. Further, with a lack of formal definitions the methodology is not defined in sufficient detail to be easily reproduced.

On the subject of related work, it is worthwhile to note that the topic of ontology matching is conceptually related to this subject, as it also involves a comparison between ontologies. Whereas matching is aimed at finding “correspondences between semantically related entities of ontologies” [16], this work is considerably different as we are attempting to compare candidate ontologies, with respect to some required semantics.

4.2 The Idea

Recall that at the core of this specification of semantic requirements is the relationship between the consequent of the entailment problem, and the models of the axiomatization.

Following from Definition 3, the superfluous models $SUP(T)$ of a candidate theory $T$ are defined as follows:

Definition 5.

\[ M \in SUP(T) \iff M \in Mod(T), M \notin \mathcal{M}^{intended} \]

In the context of reuse, superfluous models correspond to a semantic error where some aspect of the candidate is weaker than what is required. Such models satisfy the candidate, but are not models of the requirements. Such models will prevent some aspect of the requirements from being satisfied (i.e. some competency questions will not be provable) and so the theory must be strengthened if the designer wishes to reuse it. Following from Definition 4, the omitted models $OM(T)$ of a candidate theory $T$ are defined as follows:

Definition 6.

\[ M \in OM(T) \iff M \notin Mod(T), M \in \mathcal{M}^{intended} \]

On the other hand, omitted models indicate a semantic error where the theory is stronger than is required. These models are models of the requirements, $T_R$, that are not entailed by the candidate ontology. In such cases the candidate has consequences beyond what was specified in the requirements; the designer may therefore need to weaken the axioms so as to avoid this stronger semantics if it is undesirable.

Building on our understanding of semantic requirements, we can formally define a notion of preference between candidate ontologies that will serve to inform our decision. Such a definition allows for the production of unambiguous, verifiable results about the candidates. Most importantly, it simplifies the task of performing a thorough comparison, thereby reducing the investment required to obtain a deep understanding of how the candidates compare to each other, with respect to the desired ontology.

In the context of reuse, any notion of preference between candidates should address the question of which ontology is the closest match to the requirements. The underlying motivation of this being that the closer a
candidate is to the requirements, the less development work will be required to achieve the desired ontology via reuse. In the context of semantic requirements, the consideration of which semantic requirements are and are not satisfied provides an indication of the effort that will be required in each case to “bridge the gap” to obtain the desired ontology, should a particular candidate be chosen for reuse. We formalize the concept of preference as the comparative effort required between two candidates, relative to a set of semantic requirements (CQs).

### 4.2.1 Preference Defined

The concept of preference is comprised of two complimentary perspectives that capture the relative effort required to achieve the desired ontology: candidate accuracy and precision. Intuitively, an ontology that is more accurate or more precise with respect to the models of the desired ontology (as characterized by the CQs) will be preferred over other candidates; its reuse will require less effort as there are fewer corrections to be made. Both of these perspectives are naturally expressed as orderings, and are based on the possible models of the candidates, those of the CQs, and the identification of semantic errors, as described in the previous section. The orderings derived from these models represent a three-way relationship between theories – a comparison of two candidate theories, relative to the requirements.

The **Accuracy Ordering** captures the notion of required effort by extending the notion of semantic correctness discussed in the previous section; a candidate requires less effort if it is *more correct* than another, as fewer changes and corrections will need to be made in order to reuse it.

**Definition 7.** Candidate ontology \( T_2 \) is more accurate than candidate ontology \( T_1 \) (denoted by \( T_1 \preceq T_2 \)) iff

\[
SUP(T_2) \subseteq SUP(T_1), \quad OM(T_2) \subseteq OM(T_1)
\]  

(4.1)

If one candidate has fewer omitted and fewer superfluous models than another, it is clearly the more accurate candidate. In cases where one candidate does not have *both* fewer superfluous and omitted models than another, they are incomparable in the Accuracy Ordering. This is a necessary limitation because there is no means of comparing a combination of omitted and superfluous models in general.

**Lemma 1.** Candidate ontology \( T_1 \) is more accurate than candidate ontology \( T_2 \) and candidate ontology \( T_2 \) is more accurate than candidate ontology \( T_1 \) iff they are logically equivalent.

**Proof.** Candidate ontology \( T_1 \) is more accurate than candidate ontology \( T_2 \) and candidate ontology \( T_2 \) is more accurate than candidate ontology \( T_1 \) iff

\[
SUP(T_2) = SUP(T_1), \quad OM(T_2) = OM(T_1)
\]

By the definition of superfluous models, \( \mathcal{M} \in Mod(T_2) \) and \( \mathcal{M} \notin \mathcal{M}_{\text{intended}} \) iff 
\( \mathcal{M} \in Mod(T_1) \) and \( \mathcal{M} \notin \mathcal{M}_{\text{intended}} \), so that if \( \mathcal{M} \notin \mathcal{M}_{\text{intended}} \), we have \( \mathcal{M} \in Mod(T_2) \) iff \( \mathcal{M} \in Mod(T_1) \).

By the definition of omitted models, \( \mathcal{M} \notin Mod(T_2) \) and \( \mathcal{M} \in \mathcal{M}_{\text{intended}} \) iff \( \mathcal{M} \notin Mod(T_1) \) and \( \mathcal{M} \in \mathcal{M}_{\text{intended}} \), so that if \( \mathcal{M} \in \mathcal{M}_{\text{intended}} \), we have \( \mathcal{M} \in Mod(T_2) \) iff \( \mathcal{M} \in Mod(T_1) \).

Combining these cases give us \( \mathcal{M} \in Mod(T_1) \) iff \( \mathcal{M} \in Mod(T_2) \).  

In this special case of accuracy, we will say that \( T_1 \) and \( T_2 \) are equally accurate, denoted \( T_1 =_\prec T_2 \).
Lemma 2. Let $\mathcal{T}$ be a set of candidate ontologies. 
\[ \langle \mathcal{T}, \preceq \rangle \text{ is a partial ordering.} \]

Proof. If $T_1 \preceq T_2$ and $T_2 \preceq T_3$, then
\[ SUP(T_2) \subseteq SUP(T_1), OM(T_2) \subseteq OM(T_1), SUP(T_3) \subseteq SUP(T_2), OM(T_3) \subseteq OM(T_2) \]
and hence
\[ SUP(T_3) \subseteq SUP(T_1), OM(T_3) \subseteq OM(T_1) \]
so that $T_1 \preceq T_3$. Therefore $\preceq$ is transitive.

If $T_1 \preceq T_2$ and $T_2 \preceq T_1$, then
\[ SUP(T_2) \subseteq SUP(T_1), OM(T_2) \subseteq OM(T_1), SUP(T_1) \subseteq SUP(T_2), OM(T_1) \subseteq OM(T_2) \]
and hence
\[ SUP(T_2) = SUP(T_1), OM(T_2) = OM(T_1) \]
and $T_1$ and $T_2$ are equally accurate, and hence logically equivalent. Therefore $\preceq$ is antisymmetric.

Since $\preceq$ is transitive, reflexive, and antisymmetric, it is a partial ordering.

The Precision Ordering is motivated by the distinction between omitted and superfluous models – it addresses a special case in which candidates are not comparable in the Accuracy Ordering, but one should still be preferred over the other. Informally, this preference can be motivated by the observation that whether choosing an ontology for reuse or developing one from scratch, it is far easier to identify and address errors of omitted models than superfluous models. The omitted models are clearly present in the requirements theory, and the cause of their omission is unambiguously present in the theory itself. It is fairly straightforward then, to identify which axioms are causing the omission of certain models from the candidate theory (then, as part of design determine which axioms to change and how). The resolution for superfluous models on the other hand, is much less clear. Even once they are identified, the task of designing an axiom to eliminate them is more challenging as the possibilities and their implications may be complex and difficult to recognize. The Precision Ordering is defined as follows:

Definition 8. Candidate ontology $T_2$ is more precise than candidate ontology $T_1$ (denoted by $T_1 \prec T_2$) iff
\[ SUP(T_1) \neq \emptyset, SUP(T_2) = \emptyset \] (4.2)

Lemma 3. Let $\mathcal{T}$ be a set of candidate ontologies. 
\[ \langle \mathcal{T}, \prec \rangle \text{ is a strict ordering.} \]

Proof. If $T_1 \prec T_1$, then $SUP(T_1) \neq \emptyset$ and $SUP(T_1) = \emptyset$, so $\prec$ must be irreflexive.

Similarly, $T_1 \prec T_2$ and $T_2 \prec T_1$ leads to a contradiction, and $\prec$ is asymmetric.

Transitivity is trivially satisfied, since we cannot have both $T_1 \prec T_2$ and $T_2 \prec T_3$.

This second ordering may appear suspicious as it strictly prefers candidates with no superfluous models, regardless of the differences in omitted models. However, when this condition is satisfied, the relationship between the candidates’ omitted models is irrelevant to the preference between candidates. This is justified by the asymmetry between these types of models. In contrast to superfluous models, omitted models affect only the completeness
of a candidate with respect to the requirements. If we have no superfluous models, regardless of what models are omitted, we can say for certain that the models of the candidate are correct with respect to the requirements. On the other hand, any instance of superfluous models indicates that the candidate’s models are not correct with respect to the requirements and therefore it is less precise. Further, candidates without superfluous models can entail all of the CQs specified in the requirements. Assuming that the CQs do not completely characterise the desired ontology, such candidates may even be satisfactory as-is for the intended application. Thus any theory that is more precise than another will require less (possibly no) effort to reuse. So, in the context of the required effort for reuse, these more precise theories should be preferred over candidates that contain superfluous models.

Preference (\(\ll\)) is defined as the combination of the Accuracy Ordering (\(\preceq\)) and the Precision Ordering (\(<\)).

**Definition 9.** Candidate ontology \(T_2\) is preferred over candidate ontology \(T_1\) (denoted by \(T_1 \ll T_2\)) iff \(T_1 \prec T_2\) or \(T_1 \prec T_2\).

A candidate \(T_2\) is equally preferred to another candidate \(T_1\) (denoted by \(T_1 =\ll T_2\)) iff \(T_1 =\ll T_2\)

Note that we will never run the risk of having to resolve conflicting orderings to determine the Preference Ordering – if we prefer \(T_1\) over \(T_2\) then we cannot find \(T_2\) to be more correct than \(T_1\), and vice versa.

**Theorem 1.** If \(T_1 \preceq T_2\), then \(T_2 \not\ll T_1\).

*Proof.* Assume that there exist two theories, \(T_1, T_2\), such that: \(T_1 \preceq T_2\) and \(T_2 \not\ll T_1\)

If \(T_2 \ll T_1\) then, by definition we must have:

\[SUP(T_1) = \emptyset\text{ and } SUP(T_2) \neq \emptyset\]

It follows that: \(SUP(T_1) \subset SUP(T_2)\)

Therefore, by definition we cannot also have \(T_1 \preceq T_2\) as this requires that \(SUP(T_2) \subset SUP(T_1)\). Our assumption cannot hold, therefore \(\preceq\) cannot conflict with \(<\). 

In other words, if we say that \(T_2\) is more accurate than \(T_1\), then \(T_1\) cannot be more precise than \(T_2\). Since this is equivalent to saying that \(T_1 \prec T_2\) implies \(T_2 \not\preceq T_1\), we can also say that if \(T_1\) is more precise than \(T_2\) then we cannot find \(T_2\) to be more accurate than \(T_1\).

The definition of Preference allows us to formalize the fact that we prefer a more accurate and/or precise candidate as it will be less work to reuse. An analogy may be drawn between these accuracy and precision orderings and the measures of precision and recall that are often applied in the context of information retrieval. The intended models are analogous to relevant documents; whereas the precision ordering prefers those ontologies with only intended models, the information retrieval measure of precision is similar in the sense that it specifies the ratio of results that are correct (preferring a higher ratio). Similarly, the accuracy ordering is analogous to the concept of recall; recall provides a ratio of how many of the relevant results were returned, whereas a more accurate ontology captures comparatively more intended models.

If we were to consider the problem of choosing between time ontologies, using examples CQ-1 and CQ-2 from the previous chapter to approximate the intended models, the definitions would indicate that we should prefer to reuse the theory of linear points\(^1\) to that of moments\(^2\). Although both candidates are able to entail the requirements and so contain no superfluous models, the comparison would find that the theory of linear points is more accurate than the theory of moments because it omits fewer models, and is therefore closer to the required theory approximated by \(T_R\) via the CQs\(^3\).

\(^1\)http://colore.oor.net/timepoints/linear_point.clif
\(^2\)http://colore.oor.net/combined_time/moment.clif
\(^3\)Note that in practice the set of CQs would likely be much larger making for more interesting and complex results.
4.3 Viability

We now operationalize our notion of preference by introducing a set of procedures capable of producing the Preference Ordering for a given set of candidates and formalized CQs. These procedures implement a set of criteria that is provably complete and correct with respect to the definition of preference.

4.3.1 Assumptions

The procedures make the following assumptions:

- The candidates share some common (the same or overlapping) signature with the CQs. We assume whatever mappings exist between the theories have already been applied. This is a reasonable assumption due to the fact that, should a candidate be relevant in any way, there must exist mappings between its terms and those of the CQs. In fact, the procedures described to aid in the task of Search in the previous chapter produce the necessary mappings via their conjectures of relevance. These mappings must be known in order for a theory to be identified as relevant, and once they are it is straightforward to apply them to achieve a common signature. While this could be incorporated into the procedures, it is replaced with this assumption for simplicity.

- The candidates are in the same logical language as the CQs. This is reasonable given that most search tools or sources of ontologies are language-specific. Further, if a particular language is necessary or desired, it is reasonable to assume that the CQs would be formalized in this language, and the selection of candidates would include only those in the appropriate language.

- All candidate ontologies are consistent and modularized. In the absence of such an assumption there are model builders that could be employed to test this and filter out inconsistent theories; a procedure was also presented in [36] to decompose theories into modules which could be called in the case of non-modularized candidates. However, the inclusion of such procedures for every candidate would not be practical nor ideal, thus we view the modularization and consistency/verification of theories as supporting infrastructure requirements rather than necessary steps in the procedure.

4.3.2 The Procedures

The following set of procedures are designed to obtain the Preference Ordering defined in the previous section. The Preference Ordering is assigned between a pair of candidates if any of the following criteria holds:

1. If \( T_R \cup T_2 \models T_1 \) and \( \neg T_R \cup T_1 \models T_2 \) then assign \( T_2 \ll T_1 \)
2. If \( T_R \preceq T_1, T_R \not\preceq T_2 \) then assign \( T_2 \ll T_1 \)
3. If \( T_1 \equiv T_2 \) then assign \( T_1 =_{\ll} T_2 \)

Where \( T_R \) corresponds to a theory comprised of the collection of CQs, and \( T_i \) corresponds to any given candidate theory as transformed in the common, candidate hierarchy, \( H_R \) (a detailed description of this will follow). If none of the criteria is met, then a preference ordering cannot be determined, in other words one candidate is not more accurate or more precise than another and therefore they are incomparable. For any pair of theories \( T_1, T_2 \), we denote this \( T_1 \parallel T_2 \).

We adopt the concept of a hierarchy from [36], where it is presented as a means of storing similar ontologies. Formally:
**Definition 10.** A hierarchy $\mathbb{H} = \langle \mathcal{H}, \leq \rangle$ is a partially ordered, finite set of theories $\mathcal{H} = T_1, \ldots, T_n$ such that:

1. $\sigma(T_i) = \sigma(T_j)$, for all $i, j$;

2. $T_1 \leq T_2$ iff $T_2$ is an extension of $T_1$;

3. $T_1 < T_2$ iff $T_2$ is a non-conservative extension of $T_1$.

The Root Theory, $T_{\text{root}}$ of $\mathbb{H}$ is a minimal theory in the hierarchy.

We make reference to these COLORE-specific concepts in the procedures, strictly because they are convenient for the presentation of this work. It is important to note that the contributions presented here are independent of any particular repository or other approach to the search for candidate ontologies. The hierarchy is created as part of a procedure meant to provide a common basis for the candidates’ signatures in cases where the scope of the concepts in the CQs and the candidates are not completely in-line with one another. For instance, consider CQ-3 described in the example of time ontologies: its scope includes not only time but also the concept of an event. It is therefore quite possible that in this case the candidates might include both time and event ontologies. The notion of a hierarchy provides a common context to allow for the collective consideration of candidates with varying scopes that may not precisely correspond to that of the CQs ($T_{\text{R}}$).

The top-level procedure required to implement the criteria is specified below in Procedure 2. First, we consider the possible combinations between all candidates as additional candidates; this expanded set of candidates is then transferred into a common context (the Requirements’ Hierarchy, $\mathbb{H}_R$); and finally the Preference Ordering is assigned in the form of a partial order, based on the specified criteria. The role of Procedure 3 considering all possible combinations of candidates, is to identify cases where a combination of candidates may be advantageous. As in the previous example, say the CQs describe both concepts of time and events. If there are some time ontology candidates and some event ontology candidates, it may be the case that a combination of a time ontology and an event ontology will be better than any single candidate. The procedure aims to account for such cases by including candidate combinations as alternatives. Transferring theories to a common context (Procedure 4) works by taking the union of each candidate with the root theory of the Requirements Hierarchy, $\mathbb{H}_R$, which is generated with the specification of a simple tautological axiom for each term in the collective signature of candidates and requirements; the purpose of this is simply to ensure that the candidates’ signatures are comparable. After this signature expansion is complete, each theory may be organized in the hierarchy simply by determining the partial ordering via the Poset-Sort algorithm. This is accomplished according to a previously defined procedure for poset sorting from [12]. The Preference Ordering is then calculated in Procedure 5 by evaluating every combination of candidate theories against the three ordering criteria defined above.

Note that this specification of procedures is primarily intended to demonstrate the feasibility of the implementation of the Preference Ordering, and is not meant to be the most efficient approach. For example, the consideration of all possible combinations of candidates in Procedure 3 simply generates all (consistent) combinations of the candidates’ modules and adds these to the set of candidates to be considered.
Chapter 4. Candidate Preference Based on Semantic Requirements

Procedure 2 \textit{CompareCandidates}(T, T_R)

\textbf{Input:}  
1. Set of candidate theories: \( T = \{T_1, \ldots, T_n\} \)
2. \( T_R \): User-specified semantic requirements (CQs)

\textbf{Output:} A poset \( \mathcal{P} \) on all candidates and the consistent combinations between their modules, in the expanded signature \( \sigma(T_1) \cup \ldots \cup \sigma(T_n) \cup \sigma(T_R) \).

1. \( \mathcal{M} \leftarrow \text{ConsiderCombination}(T) \)
2. \( \mathcal{H} \leftarrow \text{CreateHR}(\mathcal{M}, T_R) \)
3. \( \mathcal{P} \leftarrow \text{AssignOrder}(\mathcal{H}, T_R) \)
4. Return: \( \mathcal{P} \)

Procedure 3 \textit{ConsiderCombination}(T)

\textbf{Input:} Set of candidate theories: \( T = \{T_1, \ldots, T_n\} \)

\textbf{Output:} Set \( \mathcal{M} \) of all consistent combinations between the modules of \( T_1, \ldots, T_n \)

1. \( \mathcal{M} \leftarrow \emptyset \)
2. \( \mathcal{S}_i \leftarrow \text{set of modules of } T_i \)
3. \( \mathcal{F} = \bigcup_{i=1}^{n} \mathcal{S}_i \)
4. \textbf{for each} \( G \in \wp(\mathcal{F}) \) \textbf{do}
5. \quad \text{if} \( \bigcup_{X \in G} \) is consistent \textbf{then}
6. \quad \quad \text{ADD } G \text{ to } \mathcal{M}
7. \quad \textbf{end if}
8. \textbf{end for}
9. Return: \( \mathcal{M} \)
Procedure 4 Create\(\mathbb{H}_R(M, T_R)\)

**Input:**
1. Set of theories: \(M = \{T_1, \ldots T_m\}\)
2. \(T_R\): semantic requirements (CQs)

**Output:** A hierarchy \(\mathbb{H}_R = (\mathcal{H}_R, \leq)\) such that each \(T_R, T_i^* \in \mathcal{H}_R\) is the conservative extension of each \(T_R, T_i \in M\) (respectively) by the root theory of \(\mathbb{H}_R\).

1: \(T_{\text{root}} \leftarrow \emptyset\)
2: \(\mathcal{H}_R \leftarrow \emptyset\)
3: \(\mathbb{H}_R \leftarrow (\mathcal{H}_R, \leq)\)
4: Identify the signature \(\sigma(\mathbb{H}_R)\) of the common hierarchy \(\mathbb{H}_R\) based on the union of all the unique terms in \(T_1, \ldots T_n, T_R\).
   \[
   \sigma(\mathbb{H}_R) = \sigma(T_1) \cup \ldots \cup \sigma(T_n) \cup \sigma(T_R)
   \]
5: **for each** \(n\)-ary predicate \(p_i(x_1, \ldots, x_n) \in \sigma(\mathbb{H}_R)\) **do**
   6: \(\phi_i \leftarrow (\forall x_1, \ldots, x_n)p_i(x_1, \ldots, x_n) \supset p_i(x_1, \ldots, x_n)\)
   7: \(T_{\text{root}} \leftarrow T_{\text{root}} \cup \{\phi_i\}\)
8: **end for**
9: **for each** \(n\)-ary function \(p_i(x_1, \ldots, x_n) \in \sigma(\mathbb{H}_R)\) **do**
10: \(\phi_i \leftarrow (\forall x_1, \ldots, x_n)p_i(x_1, \ldots, x_n) = p_i(x_1, \ldots, x_n)\)
11: \(T_{\text{root}} \leftarrow T_{\text{root}} \cup \{\phi_i\}\)
12: **end for**
13: \(\mathcal{H}_R \leftarrow \{T_{\text{root}}\}\)
14: **for each** \(T_i \in \{T_1, \ldots T_m, T_R\}\) **do**
15: \(T_i^* = T_i \cup T_{\text{root}}\)
16: \(\mathcal{H}_R \leftarrow \mathcal{H}_R \cup T_i^*\)
17: **end for**
18: \(\mathbb{H}_R \leftarrow \text{Poset} - \text{Sort}(\mathcal{H}_R, \leq)\)
19: **Return:** \(\mathbb{H}_R\)
Procedure 5 AssignOrder(H, TR)

Input: 1. Hierarchy of theories: H = ⟨H, ≤⟩
        2. TR: semantic requirements

Output: Poset of candidates ⟨{T1, ..., Tn}, ≪=⟩ according to the definition of Preference, with respect to TR

1: for each pair of candidate theories, Ti, Tj ∈ H, where Ti, Tj ≠ Troot do
2: if TR ≤ Tj, TR ≠ Ti then
3: Assign Ti ≪ Tj
4: else if Ti = Tj then
5: Assign Ti = ≪ Tj
6: else if TR ∪ Ti = Tj and ¬TR ∪ Tj = Ti then
7: Assign Ti ≪ Tj
8: else
9: Assign Ti ∥ Tj
10: end if
11: end for
12: Return: ⟨{T1, ..., Tn}, ≪=⟩

Algorithm Poset-Sort(P)

Input: a set P, an oracle for a poset P = (P, ≤), an upper bound on w on width of P

Output: a ChainMerge data structure for P

1: P′ ← ({e}, {}), where e ∈ P is an arbitrary theory; ▷ P′ is the current poset
2: P′ ← P \{e\}; R′ ← {};
3: U ← P \{e\}; ▷ U is the set of theories that have not been inserted
4: while U ≠ ∅ do
5: pick an arbitrary theory e ∈ U; ▷ e is the theory to be inserted in P′
6: U ← U \{e\};
7: find a chain decomposition C = {C1, C2, ..., Cq} of P′, which q ≤ w chains;
8: for i = 1, ..., q do
9: let Ci = {e_{i1}, ..., e_{il}}; where e_{i1} ≤ e_{i2}... ≤ e_{il};
10: do binary search on Ci to find the weakest theory (if any) that entails e;
11: do binary search on Ci to find the strongest theory (if any) that is entailed by e;
12: end for
13: based on the results of the binary searches, infer all relations of e with the theories in P′;
14: add into R′ all the relations of e with the elements of P′; P′ ← P′ ∪ {e};
15: P′ = (P′, R′);
16: end while
17: find a chain decomposition C of P′; build ChainMerge(P′, C) (no additional queries);
18: return ChainMerge(P′, C);

Figure 4.2: The Poset-BinInsertionSort algorithm from [12] is adapted here to determine a partial ordering of theories based on entailment.

The pseudocode for the sorting procedure is provided in Figure 4.2; it is taken directly from its original presentation and adapted to determine partial ordering is on a set of theories by searching for the “weakest/strongest theory that entails/is entailed by e”, as opposed to the smallest/largest elements. The authors employ the concept
of a chain, referring to a subset of mutually comparable elements from the ordering, in order to define and navigate the partial ordering more precisely.

**Theorem 2.** Given a set of theories, \( T = \{T_1, \ldots, T_n\} \), and a theory comprised of some semantic requirements, \( T_R \), if Procedure 2 CompareCandidates(\( T, T_R \)) terminates, it will return a poset on \( T \) and all consistent combinations of its modules, in their expanded signature \( \sigma(T_1) \cup \ldots \cup \sigma(T_n) \cup \sigma(T_R) \), according to the Preference Ordering with respect to the requirements \( T_R \).

**Proof.** CompareCandidates(\( T, T_R \)) calls procedures ConsiderCombination(\( T \)), Create\( \mathbb{H}_R(M, T_R) \), and AssignOrder(\( \mathbb{H}, T_R \)). Therefore, if we show that:

- \( \text{AssignOrder}(\mathbb{H}, T_R) \) returns a poset on all of the theories in \( \mathbb{H} \) according to the Preference ordering, with respect to \( T_R \);
- and \( \text{Create}\( \mathbb{H}_R(M, T_R) \) creates a hierarchy containing all of the theories in \( M \) and \( T_R \) and all consistent combinations of their modules, in their expanded signature;
- and \( \text{ConsiderCombination}(T) \) generates \( T \) and all consistent combinations of its modules,

we will have shown that Procedure 2 CompareCandidates(\( T, T_R \)) returns a poset on \( T \) and all consistent combinations of its modules, in their expanded signature \( \sigma(T_1) \cup \ldots \cup \sigma(T_n) \cup \sigma(T_R) \), according to the Preference Ordering with respect to the requirements \( T_R \).

**Claim:** given some set of theories \( T = \{T_1, \ldots, T_n\} \), if Procedure 3 ConsiderCombination(\( T \)) terminates, it will return all consistent combinations of the modules of \( T \).

This follows from the definition of a power set. By definition, the power set of \( F \) is the set of all subsets of \( F \); in other words, the set of all sets of the modules of the theories in \( T \). For each member of the set, the procedure evaluates whether the union of its members (modules) is consistent (line 5) and any consistent members are added to the set \( M \) to be returned. Thus \( M \) contains all consistent combinations of the modules of \( T \).

**Claim:** given some set of theories \( M \) and some theory \( T_R \), if Procedure 4 Create\( \mathbb{H}_R(M, T_R) \) terminates it will return a hierarchy \( \mathbb{H}_R = \langle \mathcal{H}_R, \leq \rangle \) such that each \( T_R, T_i \in \mathcal{H}_R \) is the conservative extension of each \( T_R, T_i \in M \), (respectively) by the root theory of \( \mathbb{H}_R \).

Lines 1-3 of the procedures initialize the hierarchy \( \mathbb{H}_R \).

Lines 4-13 create the root theory \( T_{\text{root}} \) of \( \mathbb{H}_R \).

Lines 14-17 adds \( T_{\text{root}} \) to each \( T_i \in M \), and to \( T_R \), and adds the extension to \( \mathcal{H}_R \). Therefore, the partially ordered set of theories, sorted on entailment (as per the algorithm from [12]) returned on line 18 is in fact a hierarchy \( \mathbb{H}_R = \langle \mathcal{H}_R, \leq \rangle \) such that each \( T_R, T_i \in \mathcal{H}_R \) is the conservative extension of each \( T_R, T_i \in M \), (respectively) by the root theory of \( \mathbb{H}_R \).

Since \( T_{\text{root}} \) is composed solely of tautologies, each \( T_i, T_R \) must be conservative extensions of each \( T_i, T_R \) (respectively) in the expanded signature \( \sigma(T_1) \cup \ldots \cup \sigma(T_n) \cup \sigma(T_R) \).

**Claim:** given a hierarchy \( \mathbb{H} = \langle \mathcal{H}, \leq \rangle \) and a theory \( T_R \), if Procedure 5 AssignOrder(\( \mathbb{H}, T_R \)) terminates, it will return a poset on the theories in \( \mathcal{H} \), according to the Preference Ordering with respect to \( T_R \).

The procedure considers each pair of theories, \( T_i, T_j \in \mathcal{H} \), with the exception of the requirements \( T_R \) and the root theory \( T_{\text{root}} \). It evaluates each pair against the Preference Ordering Criteria 1, 2, and 3 on lines 6, 2, and 4 (respectively). According to Theorem 1, the criteria do not conflict, so the ordering of their assessment is irrelevant. The completeness and correctness of Criteria 3 for equally preferred candidates is a result of Lemma 1.
of Theorem 4, proven in Section 4.4. Thus the procedure will assign the correct Preference Ordering for each pair of candidates in $\mathcal{H}$ (with the exception of $T_R$ and $T_{\text{root}}$). By Lemma 2 and Lemma 3, this ordering will be a poset.

`CompareCandidates(T, T_R)` first calls `ConsiderCombination(T)`, which we have shown will return all consistent combinations of the modules of $T$. It then calls `CreateH_R(M, T_R)`, where $M$ is the set of theories returned by `CompareCandidates(T, T_R)`. We have shown this returns a hierarchy $H_R = \langle H_R, \leq \rangle$ containing all candidates and the consistent combinations of their modules in the expanded signature $\sigma(T_1) \cup \ldots \cup \sigma(T_n) \cup \sigma(T_R)$. Finally, `AssignOrder(H, T_R)` is called, where $H$ is the hierarchy returned by `CreateH_R(M, T_R)`. We have shown this will return a poset, according to the Preference Ordering over all of the theories in $H$ with respect to the requirements, $T_R$.

4.3.3 An Example: Choosing Between Participation Ontologies

To demonstrate more concretely how this process would work in practice, we now turn to a small example. Consider some scenario in which we would like to reuse an ontology to axiomatize the notion of participation. Assume the following CQs represent our current knowledge and understanding of the semantic requirements. When formalized, we prefix all terms with $t_r$ to distinguish them from any like-labelled terms in the candidates’ signatures.

1. Anything that participates in some event has some role when they are participating.

   $$\forall x, a, t \; t_r\text{-participates\_in}(x, a, t) \land t_r\text{-event}(a) \supset (\exists z) t_r\text{-has\_role}(x, z, t)$$

2. An object must exist in order for it to participate in some event.

   $$\forall x, a, t \; t_r\text{-participates\_in}(x, a, t) \land t_r\text{-event}(a) \supset t_r\text{-exists\_at}(x, t)$$

3. If something participates in an activity, then this participation translates to any other event that the activity may be a part of.

   $$\forall x, a_1, a_2, t \; t_r\text{-participates\_in}(x, a_1, t) \land t_r\text{-part\_of}(a_1, a_2) \supset t_r\text{-participates\_in}(x, a_2, t)$$

4. An object only participates in a single activity at a particular time.

   $$\forall x, a_1, a_2 \; t_r\text{-object}(x) \land t_r\text{-participates\_in}(x, a_1, t) \land t_r\text{-participates\_in}(x, a_2, t) \supset a_1 = a_2 \lor t_r\text{-part\_of}(a_1, a_2)$$

5. We need to talk about different types of participation: active, passive, consumption.

   $$\forall x, a, t \; t_r\text{-participates\_in}(x, a, t) \supset t_r\text{-active\_in}(x, a, t) \lor t_r\text{-passive\_in}(x, a, t) \lor t_r\text{-consumed\_in}(x, a, t)$$

This collection of CQs forms the theory of requirements, denoted $T_R$. 
Let us now assume that our search efforts have uncovered the following ontologies which we deem potential candidates for reuse: PSL\_participates\(^4\) DOLCE\_participation\(^5\) Gangemi\_participation.clif\(^6\) No additional knowledge about these theories is required to produce the Preference Ordering between them.

**Conjectures of Relevance**

Manually, or via an assumed search procedure, we make a set of conjectures of relevance for the terms in each theory, in the form of mapping axioms from the signature of the candidate theory to the signature of the required theory, \(Th(\mathcal{M}_{\text{intended}})\), which is currently known only approximately by the set of CQs. The following translation definitions represent the conjectured relationship between the signatures of the candidate ontologies and the terms used in the specification of our semantic requirements. In other words, these perceived mappings between the signatures and the CQs represent the conjectures of relevance discussed in Chapter 3. In practice, we assume the use of some automated procedure to will supply such results, though they will possibly be further augmented by user input.

**Relevance of PSL\_participates** \(\Delta_{PSL\_participates \rightarrow TR}\)

- \((\forall x,a,t)\text{participates}\_in(x,a,t) \iff tr\_participates\_in(x,a,t)\)
- \((\forall a)\text{activity}\_occurrence(a) \iff tr\_event(a)\)
- \((\forall a)\text{exists}\_at(a) \iff tr\_exists\_at(a)\)
- \((\forall a_1,a_2)\text{subactivity}\_occurrence(a_1,a_2) \iff \text{part}\_of(a_1,a_2)\)

**Relevance of DOLCE\_participation** \(\Delta_{DOLCE\_participation \rightarrow TR}\)

- \((\forall x,a,t)PC(x,a,t) \iff tr\_participates\_in(x,a,t)\)
- \((\forall a)PD(a) \iff tr\_event(a)\)
- \((\forall x,t)PRE(x,t) \iff tr\_exists\_at(x,t)\)

**Relevance of gangemi\_participation** \(\Delta_{gangemi\_participation \rightarrow TR}\)

- \((\forall x)\text{Event}(x) \iff tr\_event(x)\)
- \((\forall x)\text{Object}(x) \iff tr\_object(x)\)

---

\(^4\)A module of the PSL ontology [32] [http://colore.oor.net/psl_participates/psl_participates.clif](http://colore.oor.net/psl_participates/psl_participates.clif)

\(^5\)A module of the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [22] [http://colore.oor.net/dolce_participation/dolce_participation.clif](http://colore.oor.net/dolce_participation/dolce_participation.clif)

\(^6\)A set of OWL axioms provided by Aldo Gangemi [21] [http://colore.oor.net/gangemi-participation/gangemi.clif](http://colore.oor.net/gangemi-participation/gangemi.clif)
•

\[(\forall x, y)\text{hasParticipant}(x, y) \iff (\exists t)\text{tr_participates_in}(y, x, t)\]

•

\[(\forall x, y)\text{pre}(x, y) \iff \text{tr_exists_at}(x, t)\]

**Creation of a Common Hierarchy**

In the specified procedure, prior to the creation of the candidate hierarchy, the procedure would consider potential combinations of candidates. However given that each of the three candidates are single modules, focused on the same domain, (and in the interests of practicality) we omit the consideration of useful combinations here.

To create the common hierarchy requires both the application of relevance conjectures to obtain some common signature with \(T_r\), as well as the expansion of the signatures of each theory, as required such that all theories have the same signature. We generate tautological axioms for each of the terms in the candidates’ and requirement’s signatures, as described in the procedure. For example, for the term \(\text{tr.active.in}\) (for which there are no mappings from any of the candidates’ signatures), we add the axiom:

\[(\forall x, a, t)\text{tr.active.in}(x, a, t) \supset \text{tr.active.in}(x, a, t)\]

These axioms are created and added to the theories for each term. In the case of a function, \(\text{beginof}(x)\) for example, we add tautological axioms of the form:

\[(\forall x)\text{beginof}(x) = \text{beginof}(x)\]

The collection of all of these tautologies forms the root theory for the common hierarchy, we denote this \(H_{\text{root}}\).

This result of this procedure is a new version of each of the input theories, that has an equivalent signature with all of the other candidates, as well as the requirements. We’ll refer to these revised theories using the prefix \(H_{\text{.}}\). Note that while we could apply the mappings to each theory to derive theories with a (partially) common signature to \(T_r\), for the purposes of this example we opt to simply add the translation definitions as part of the same theory, for the same effect.

\[
H_{\text{PSL.participates}} = \text{PSL.participates} \cup \Delta_{\text{psl} \rightarrow T_r} \cup H_{\text{root}}
\]

\[
H_{\text{DOLCE.participation}} = \text{DOLCE.participation} \cup \Delta_{\text{DOLCE} \rightarrow T_r} \cup H_{\text{root}}
\]

\[
H_{\text{gangemi.participation}} = \text{gangemi.participation} \cup \Delta_{\text{gangemi} \rightarrow T_r} \cup H_{\text{root}}
\]

And finally, we expand the theory of the requirements:

\[
H_{\text{T_r}} = T_r \cup H_{\text{root}}
\]

We know that each theory will extend the root theory, \(H_{\text{root}}\), but to complete the addition of each of these theories into the common hierarchy, we must assess whether there exist any partial orderings between them (i.e. the \(\leq\) relationship). As discussed, this sorting procedure requires the evaluation of a set of entailment problems. For any \(T_i, T_j\) in the candidate hierarchy, if we determine \(T_i \models T_j\) then we can assign \(T_j \leq T_i\) in the hierarchy ordering \(H_{\leq}\). All files generated to determine the results below can be found in Appendix B.
- \( H_{PSL} \text{participates} \models H_{TR} \)  
  no proof; counter-example found for CQ1,  
  therefore: \( H_{PSL} \text{participates} \not\models H_{TR} \)

- \( H_{PSL} \text{participates} \models H_{gangemi} \text{participation} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{PSL} \text{participates} \not\models H_{gangemi} \text{participation} \)

- \( H_{PSL} \text{participates} \models H_{DOLCE} \text{participation} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{PSL} \text{participates} \not\models H_{DOLCE} \text{participation} \)

- \( H_{gangemi} \text{participation} \models H_{TR} \)  
  no proof; counter-example found for CQ1,  
  therefore: \( H_{gangemi} \text{participation} \not\models H_{TR} \)

- \( H_{gangemi} \text{participation} \models H_{DOLCE} \text{participation} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{gangemi} \text{participation} \not\models H_{DOLCE} \text{participation} \)

- \( H_{gangemi} \text{participation} \models H_{PSL} \text{participates} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{gangemi} \text{participation} \not\models H_{PSL} \text{participates} \)

- \( H_{DOLCE} \text{participation} \models H_{TR} \)  
  no proof; counter-example found for CQ1,  
  therefore: \( H_{DOLCE} \text{participation} \not\models H_{TR} \)

- \( H_{DOLCE} \text{participation} \models H_{PSL} \text{participates} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{DOLCE} \text{participation} \not\models H_{PSL} \text{participates} \)

- \( H_{DOLCE} \text{participation} \models H_{gangemi} \text{participation} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{DOLCE} \text{participation} \not\models H_{gangemi} \text{participation} \)

- \( H_{TR} \models H_{PSL} \text{participates} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{TR} \not\models H_{PSL} \text{participates} \)

- \( H_{TR} \models H_{gangemi} \text{participation} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{TR} \not\models H_{gangemi} \text{participation} \)

- \( H_{TR} \models H_{DOLCE} \text{participation} \)  
  no proof; counter-example found for ax1,  
  therefore: \( H_{TR} \not\models H_{DOLCE} \text{participation} \)
Evaluation of Criteria

Once the theories have been added to the common hierarchy, we can begin the evaluation of the Preference Criteria, as in Procedure 5. The first criteria (line 2) in Procedure 5 can be observed directly from the hierarchy.

For each pair of candidates $T_i, T_j$ if $T_R \leq T_j, T_R \not\leq T_i \rightarrow T_i \ll T_j$:

Based on the hierarchy orderings determined above, we are unable to infer any Preference Orderings.

The second criteria (line 4) clearly cannot hold, since equality between any theories would have been shown by the entailment results in the previous assessment.

Finally, we examine the final criteria (line 6) of Procedure 5. For each pair of candidates, $T_i, T_j$ we must assess whether $T_i \cup T_R \models T_R \models \Delta_{gangemi \rightarrow T_R}$ and $\neg T_R \cup T_j \models T_i$. Since no ordering has been identified between the candidates due to the hierarchy, we must assess this criteria for each pair of candidates.

$T_i = PSL_{participates}, T_j = gangemi_{participation} : PSL_{participates} \cup T_R \cup \Delta_{gangemi \rightarrow T_R} \models gangemi_{participation}$

No proof; counterexample found for ax1, therefore: $PSL_{participates} \cup T_R \cup \Delta_{gangemi \rightarrow T_R} \not\models gangemi_{participation}$. The criteria cannot be met in this case, so we do not need to evaluate the second entailment problem and we can infer that $PSL_{participates} \parallel gangemi_{participation}$.

$T_i = gangemi_{participation}, T_j = PSL_{participates} : gangemi_{participation} \cup T_R \cup \Delta_{PSL \rightarrow T_R} \models PSL_{participates}$

No proof; counterexample found for ax1, therefore: $gangemi_{participation} \cup T_R \cup \Delta_{PSL \rightarrow T_R} \not\models PSL_{participates}$. The criteria cannot be met in this case, so we do not need to evaluate the second entailment problem and we can infer that $gangemi_{participation} \parallel PSL_{participates}$.

$T_i = DOLCE_{participation}, T_j = PSL_{participates} : DOLCE_{participation} \cup T_R \models PSL_{participates}$

No proof; counterexample found for ax1, therefore: $DOLCE_{participation} \cup T_R \not\models PSL_{participates}$. The criteria cannot be met in this case, so we do not need to evaluate the second entailment problem and we can infer that $DOLCE_{participation} \parallel PSL_{participates}$.

$T_i = PSL_{participates}, T_j = DOLCE_{participation} : PSL_{participates} \cup T_R \models DOLCE_{participation}$

No proof; counterexample found for ax1, therefore: $PSL_{participates} \cup T_R \not\models DOLCE_{participation}$. The criteria cannot be met in this case, so we do not need to evaluate the second entailment problem and we can infer that $PSL_{participates} \parallel DOLCE_{participation}$.

$T_i = DOLCE_{participation}, T_j = gangemi_{participation} : DOLCE_{participation} \cup T_R \models gangemi_{participation}$

No proof; counterexample found for ax1, therefore: $DOLCE_{participation} \cup T_R \not\models gangemi_{participation}$. The criteria cannot be met in this case, so we do not need to evaluate the second entailment problem and we can infer that $DOLCE_{participation} \parallel gangemi_{participation}$.

$T_i = gangemi_{participation}, T_j = DOLCE_{participation} : gangemi_{participation} \cup T_R \models DOLCE_{participation}$

No proof; counterexample found for ax1, therefore: $gangemi_{participation} \cup T_R \not\models DOLCE_{participation}$. The criteria cannot be met in this case, so we do not need to evaluate the second entailment problem and we can infer that $gangemi_{participation} \parallel DOLCE_{participation}$. 
Results

In summary, evaluation of the criteria has indicated that the three candidate ontologies are incomparable in terms of preference, with respect to the semantic requirements. Note that while it would have been straightforward to construct a set of CQs for which one candidate would clearly be preferred, such an approach would have been artificial. Further, it might have served as an oversimplification of the case of choice (i.e. where there is a clear best alternative) and thus undermined the work presented here. We emphasize that the purpose of this example is simply to illustrate more concretely how the procedures presented in this section would work in practice.

Further analysis of the proofs generated and the counterexamples found may be performed to gain a deeper insight into the nature of the models motivating the results. However, such a need will likely only arise in the case where two candidates are closely matched in all other assessments as well as their Preference Orderings, or in the case that the results of the ordering are somehow contrary to what was expected. In any case, previous work on the use of automated theorem provers for the verification of ontologies [50] may be leveraged to support such analysis. This work examines the various possible outcomes of entailment problem evaluation (originally, for ontology verification via CQs), and the analysis of issues of ‘unintended proofs’ as well as failure to prove certain sentences. Since the assessment of each of the criteria amounts to no more than the evaluation of an entailment problem, the same approach may be applied here to address any suspicion or confusion regarding the proof results.

It is interesting to recognize that had we considered the candidates’ CQ results individually, we would be likely to draw the conclusion that DOLCE’s theory of participation is the most suitable candidate as it satisfies comparatively more (albeit only one) CQs than the other ontologies. However, the assessment performed here illustrates that this is not the case; in fact, this example demonstrates how the isolated CQ results may be misleading with respect to the concept of preference between candidate ontologies. Despite satisfying one of the CQs, DOLCE_participation is neither more precise nor more accurate than the other ontologies. This demonstrates the value of having a procedure to determine the preference ordering between candidate ontologies; while the concept of preference is natural and intuitive, the ordering itself may not necessarily be readily apparent.

4.4 Correctness and Completeness of Criteria

From Lemma[1] two candidates are equally accurate if and only if they are logically equivalent. This addresses both the correctness and the completeness of the third criteria in the procedure. In the following, we consider the first two criteria, (1) $T_R \cup T_2 \models T_1$ and $\neg T_R \cup T_1 \models T_2$, and (2) $T_R \leq T_1, T_R \not\preceq T_2$ that are used in the previous procedure to detect and assign a preference ordering between candidates.

Theorem 3. Suppose $T_R$ corresponds to a theory comprised of the collection of competency questions. For any two candidate ontologies $T_1, T_2$.

$T_2 \preceq T_1$ iff:

1. $T_R \cup T_2 \models T_1$ and $\neg T_R \cup T_1 \models T_2$, or

2. $T_R \leq T_1, T_R \not\preceq T_2$

Proof. $\Leftarrow$

1. If $T_R \cup T_2 \models T_1$ and $\neg T_R \cup T_1 \models T_2$ then $T_2 \preceq T_1$

---

7The individual evaluation of the candidates against each CQ is provided in Appendix B.
CHAPTER 4. CANDIDATE PREFERENCE BASED ON SEMANTIC REQUIREMENTS

Proof. If \( T_R \cup T_2 \models T_1 \), then:

For all models \( \mathcal{M} \):

\[
\mathcal{M} \in \text{Mod}(T_R), \mathcal{M} \in \text{Mod}(T_2) \rightarrow \mathcal{M} \in \text{Mod}(T_1)
\]

\[
\mathcal{M} \in \text{Mod}(T_R), \mathcal{M} \notin \text{Mod}(T_1) \rightarrow \mathcal{M} \notin \text{Mod}(T_2)
\]

By definition:

\[
OM(T_1) \subseteq OM(T_2)
\]

If \( \neg T_R \cup T_1 \models T_2 \) then:

For all models \( \mathcal{M} \):

\[
\mathcal{M} \notin \text{Mod}(T_R), \mathcal{M} \in \text{Mod}(T_1) \rightarrow \mathcal{M} \in \text{Mod}(T_2)
\]

By definition:

\[
SUP(T_1) \subseteq SUP(T_2)
\]

Therefore, if \( T_R \models T_2 \supset T_1 \) and \( \neg T_R \models T_1 \supset T_2 \) then by definition we must have

\[
OM(T_1) \subseteq OM(T_2) \quad \text{and} \quad SUP(T_1) \subseteq SUP(T_2)
\]

Therefore, \( T_2 \preceq T_1 \) implies \( T_2 \ll T_1 \).

2. If \( T_R \leq T_1, T_R \nleq T_2 \) then assign \( T_2 \ll T_1 \)

Proof. Since \( T_R \leq T_1 \rightarrow \text{Mod}(T_R) \subseteq \text{Mod}(T_1) \).

By definition: \( SUP(T_1) = \emptyset \).

Since \( T_R \nleq T_2 \rightarrow \text{Mod}(T_R) \nsubseteq \text{Mod}(T_1) \).

By definition: \( SUP(T_2) \neq \emptyset \).

Therefore \( T_2 \prec T_1 \) implies \( T_2 \ll T_1 \).

3. If \( T_1 \equiv T_2 \) then assign \( T_1 = \ll T_2 \)

Proof. If \( T_1 \equiv T_2 \), then trivially: \( SUP(T_1) = SUP(T_2), OM(T_1) = OM(T_2) \).

Therefore \( T_1 = \ll T_2 \) implies \( T_1 = \ll T_2 \).

⇒

If none of the conditions are met, then the Preference Ordering cannot be determined, in other words one candidate is not more accurate or more precise than another and therefore they are incomparable. For any pair of theories \( T_1, T_2 \), we denote this \( T_1 \parallel T_2 \).

For all \( T_1, T_2 \) in candidate hierarchy \( \mathbb{H}_R \), if \( T_1, T_2 \) are comparable, (i.e. \( T_1 \ll T_2 \) or \( T_2 \ll T_1 \)) then at least one of the procedure’s criteria will be met.

Assume that a Preference Ordering holds between two candidates, \( T_1, T_2 \) and none of the criteria hold. By definition, if \( T_1 \ll T_2 \), then either \( T_1 \preceq T_2 \) or \( T_1 \prec T_2 \) must be true.

Assume \( T_1 \prec T_2 \): By definition,

\[
T_1 \preceq T_2 \iff SUP(T_2) \subseteq SUP(T_1), OM(T_2) \subseteq OM(T_1)
\]
For any two candidate theories, $T_1, T_2$ in the Candidate Hierarchy, $H_R$, from the definition of $OM(T)$:

$$OM(T_1) \subseteq OM(T_2) \iff (M \in Mod(T_R), M \notin Mod(T_1)) \rightarrow M \notin Mod(T_2)) \iff M \in Mod(T_R), M \in Mod(T_2) \rightarrow M \in Mod(T_1) \iff T_R \cup T_2 \models T_1$$

For any two candidate theories, $T_1, T_2$ in the Candidate Hierarchy, $H_R$, from the definition of $SUP(T)$:

$$SUP(T_1) \subseteq SUP(T_2) \iff (M \in Mod(T_1), M \notin Mod(T_R)) \rightarrow M \in Mod(T_2)) \iff T_1 \cup \neg T_R \models T_2$$

This conflicts with our assumption that no criteria hold, as we have shown it is impossible to have $T_1 \preceq T_2$ and not satisfy Criteria 1.

**Assume** $T_1 \prec T_2$: By definition,

$$T_1 \prec T_2 \iff SUP(T_1) \neq \emptyset, SUP(T_2) = \emptyset$$

By definition of $SUP(T)$, if:

$$SUP(T_2) = \emptyset$$

$$M \in Mod(T_R) \rightarrow M \in Mod(T_2)$$

$$T_R \leq T_2$$

If

$$SUP(T_1) \neq \emptyset$$

There exists some

$$M \text{ s.t. } M \notin Mod(T_R), M \in Mod(T_1)$$

$$T_R \not\leq T_1$$

This conflicts with our assumption as we have shown that it is impossible to have $T_1 \prec T_2$ and not satisfy Criteria 3.

4.5 **Discussion**

The reader will likely have noticed that the transfer to a hierarchy structure in Procedure 4 is not necessary to determine the Preference ordering. The motivation for its inclusion comes, not only for convenience of presentation, but from a more general objective of encouraging reuse. The hierarchy structure provides useful means of storing the candidates in a common context after the translation and expansion of their signatures. It also offers a potential, structured venue for storing and reusing results of these evaluations. Beyond this, a procedure was
defined in [36] to insert theories into the correct position in a hierarchy and infer relationships such as the similarities and differences between them. Such information might provide valuable insight for the task of choice, and the use of the hierarchy structure supports this potential extension to the procedure presented here. For reference, this procedure is supplied in Appendix A. We identify and discuss other decisions that were made in the design of these procedures in the following.

4.5.1 The Sufficiency of CQs

While the use of CQs as requirements is well-established, their role as approximating the required theory, \( T_R \approx \text{Th}(\text{intended}) \), in the implementation of the Preference Ordering may be subject to speculation. However at this stage in development, no better approximation of the theory exists. Further, if some more detailed draft axioms or similar information does exist, these sentences could be added to \( T_R \) without affecting the definition or the procedures.

In the review of related work, we saw approaches that assigned weightings or priorities to the requirements. It follows from this perspective that requirements are often times not equally important, and thus our definition of preference may be criticized as lacking this dimension. However, it is our position that in the context of semantic requirements, such approaches are flawed; given that these are semantic requirements, there should be nothing malleable about them. Although the decision of which interpretations should be models of the ontology may be subject to revision, at any point in time a model is either desired or it is not.

4.5.2 \( H_R \) Root Theory Definition

A reader familiar with the notion of a hierarchy as defined in COLORE will have noticed that the root theory used here differs from that which is defined in the COLORE literature [36]. This deviation was supported by pragmatic and philosophical factors; we discuss the alternative approaches, and the rationale behind this decision here.

Typically, the root theory represents the most basic ontological commitment in a given hierarchy (domain). In the context of COLORE, where hierarchies provide a tool to organize different theories, it makes sense to require that a hierarchy have a unique root theory. This acts as a kind of sorting mechanism as it requires that theories with distinct ontological commitments be stored in separate hierarchies.

For our procedure, we wanted to gather the candidate theories into a single hierarchy for comparison, but what would it mean to be the root theory of such a hierarchy? After all, a hierarchy of candidate theories is not necessarily a true hierarchy; it is a collection of candidate conjectures gathered via some heuristics, but there is no guarantee that there is a meaningful, underlying commitment shared between them. In fact we expect their underlying commitments will often differ. The root theory was a necessary tool to expand the signature of the candidate theories (and semantic requirement theory) in a consistent manner, such that they could be considered in the same hierarchy. The concept of the root theory for this task was appealing in that, as the ‘weakest theory’ in the hierarchy it would be a benign addition to each of the theories in order to achieve the required vocabulary. However, as we have already noted - the meaning and attainability of a true root theory in a scenario of varied candidates with overlapping and possibly even disjoint signatures is unclear.

One potential means of discovering the root theory was to take an informed approach, requiring the identification and consideration of the root of each candidate. The idea would be to iteratively create a root theory for \( H_R \), starting with the root theory of the requirements, and revising it by finding the similarity (the weakest common theory, as defined in [36]) between itself and one of the candidate theories. Several potential questions arise when considering this case, such as: What would be the initial root theory? What is the root of the requirements theory?
How would we generate a new theory when there is some signature overlap? The signature mismatch would need to be resolved before being able to find the similarity, however the original purpose of the root theory was to provide the candidates with a common signature. If we were able to solve the issue of mismatched signatures in order to identify the similarities between the candidates, then we have no need for the root theory! While it still may be interesting or useful to identify the ‘true’ root theory of the candidates’ hierarchy, finding the similarity between two theories is theoretically possible but it is also complicated and would need to be performed a number of times. Would this approach even be scalable?

Instead, we chose to derive the weakest possible root for $\mathbb{H}_R$ based solely on the signature of $T_R$ and all of the candidates. To maintain simplicity, this approach would generate a simple axiom for each term in the signature which would introduce no semantics, i.e. a tautology, (this process is described in full in Procedure 4). This approach appeals to any pragmatic concerns as the necessary implementation is straightforward, presenting no feasibility or scalability concerns. It also offers a completely uniform solution in that the axioms will be generated, without exception, with the same intuition and the same method; this is crucial to avoid skewing the assessment results. Finally, when considering the meaning of the hierarchy in this case, a root theory of tautologies appeals to our intuitions. As opposed to traditional hierarchies that group theories based on some shared, ontological commitment, the purpose of this hierarchy is to provide a common context for comparison of the candidates. In this case we cannot assume there exists any meaningful, shared ontological commitments between all of the candidates and the requirements. It is logical then, that the shared commitment between any set of candidate consists of a theory comprising the collective signature, but with essentially meaningless axioms.

### 4.5.3 Implementation Concerns

As mentioned, the procedure presented here is a bare-bones demonstration of feasibility; as such it does not attempt any clever approaches to implementation. While in theory these procedures may be followed and even (semi-)automated to achieve the desired results, in their current state they would not make for a very pragmatic solution.

For example, in Procedure 3 we provide for a consideration of potential candidate combinations through the addition of all possible combinations of candidate modules. This is a brute force approach – although it will identify combinations of candidates that may be advantageous, it will also result in the consideration of many combinations which might easily be identified as useless. For example, any combination of two candidates is rather pointless if it does not improve the standing of both theories. Consider the case where two candidates, $T_1, T_2$ have complimentary signatures: if we combine $T_1$ and $T_2$ then the resulting theory (a new candidate, let’s say $T_3$) would have a broader scope than both $T_1$ and $T_2$ and thus potentially be more capable in satisfying the semantic requirements. On the other hand, if $T_2$’s signature contained the signature of $T_1$, then in terms of scope we are no better off with $T_3$ than we were with $T_2$. It would be straightforward enough to implement a procedure that captures this observation in order avoid the unnecessary assessment of these sorts of non-beneficial combinations. For example, rather than adding all combinations of modules, we could perform a simple check to determine whether the combination would be advantageous in terms of signature. If the combination of some theories has a broader scope than both original theories (i.e. if $\sigma(T_3) \supset \sigma(T_1)$ and $\sigma(T_3) \supset \sigma(T_2)$) then it would be added to the set of candidates to consider for reuse. If this condition is not met then, with respect to breadth of scope, there is no reason to consider it. This sort of approach, outlined in Procedure 7, could take the place of the brute force approach presented previously.

A similar observation can be made with respect to the candidates’ semantics; if the combination of two theories does not result in a new candidate that is stronger (potentially better) with respect to the semantic requirements,
Procedure 7 ScopeBeneficialCombination(\(T\))

**Input:** \(T\): Set of \(n\) candidates theories: \(T_1, ...T_n\), each comprised of at least one module; where \(S_{ij}\) denotes the \(j^{th}\) module of the \(i^{th}\) theory.

**Output:** \(M_{\text{scope}}\): Set of all combinations between \(T_1, ...T_n\) that are potentially beneficial for scope.

1. for all combinations of \(S_{ij}\)'s do
2. if \(S_{ak} \cup ... \cup ...S_{bp}\) mutually consistent AND \(\sigma(S_{ak} \cup ... \cup ...S_{bp}) \supset \sigma(S_{ak}), ...\sigma(S_{bp})\) then
3. ADD \(S_{ak} \cup ... \cup ...S_{bp}\) to \(M_{\text{scope}}\)
4. end if
5. end for

then we should not consider it. This could be accounted for with a similar approach, as outlined in Procedure 8. Alternatively, we might simply omit consideration of combinations altogether unless prompted by the designer.

Procedure 8 SemanticsBeneficialCombination(\(T\))

**Input:** \(T\) Set of \(n\) candidates theories: \(T_1, ...T_n\), each comprised of at least one module; we adopt the following naming convention for theories’ modules: \(S_{ij}\) is the \(j^{th}\) module of the \(i^{th}\) theory.

**Output:** \(M_{\text{semantics}}\): Set of all combinations between \(T_1, ...T_n\) that are potentially beneficial for scope.

1. for all combinations of \(S_{ij}\)'s... do
2. if \(S_{ak} \cup ... \cup ...S_{bp}\) mutually consistent AND \(S_{ak},...,S_{bp}\) \(R_{\subset}= (S_{ak} \cup ... \cup ...S_{bp})\) then
3. ADD \(S_{ak} \cup ... \cup ...S_{bp}\) to \(M_{\text{semantics}}\)
4. end if
5. end for

Beyond these suggestions, there is a great deal of room to improve upon the design of the procedures we have presented here. In general, while we recognize that the implementation’s efficiency is an important consideration, it is out of the scope of this work.

4.6 Summary

By leveraging the formal and objective nature of CQs we were able design a procedure capable of producing a useful comparison between candidate ontologies. We have provided an implementable solution that successfully simplifies the evaluation and assessment of candidate ontologies with respect to the semantic requirements, and identified several directions for improvement and future work. With these procedures, it is straightforward to see how the assessment of semantic requirements may be offloaded to an independent resource, thus contributing to the minimization of the developer’s workload presented in the previous chapter. While considerably beyond the scope of this thesis, it will also be interesting to consider other applications of this notion of preference, such as in the related areas of software reuse or the task of assessing ontology shareability.
Chapter 5

Attaining the Benefits of Reuse

A major issue with the current state of reuse is that its supposed benefits are unreliable. The additional work required for design via reuse is motivated only by promises that it will ‘pay off’ via the benefits of reuse. Developers are told to expect not only a reduced design workload, but ‘free’ semantic interoperability with the reused theories along with any other ontologies that have also reused them. In reality these benefits are rarely seen, thus they provide little motivation to design ontologies via reuse. The second objective of this thesis is for the benefits of reuse to result predictably. In other words, the developer should be able to determine what benefits will or won’t result form a particular instance of reuse, as well as how their decisions may impact this outcome.

The idea that these benefits are a consequence of reuse is a misconception; this is clearly based on the current state where many instances of reuse fail to achieve one or both of the expected benefits. Rather than attempting to ensure these benefits can be guaranteed unconditionally, the focus of this objective is on the guarantee itself; by making the benefits more explicit and transparent, the developer is provided with the ability to determine how their choices impact the potential benefits of reuse. To achieve this, it is necessary that the concept of reuse itself is defined explicitly. In contrast to the previous chapters, here we are concerned specifically with the actual reuse of a particular ontology to satisfy the requirements. What kinds of manipulations can be performed when reusing an ontology, and what falls outside of the scope? Perhaps more importantly, with what criteria is the boundary of the scope of reuse drawn?

We extend the intuitions presented previously to develop a definition for the task of reuse. In ontology design, we are attempting to create a set of axioms that captures some intended semantics of a set of concepts. Reuse is often conceptualized as a special case of design; intuitively, it refers to the task of taking some existing ontology(s) and manipulating it in some way in order to satisfy the design requirements. This observation is supported by existing, informal definitions of reuse. In addition, there is an implicit condition which is often not stated as it is perhaps assumed as common sense – reuse can only be performed on a reusable ontology. To illustrate this with an example, consider the development of a finance ontology. It would be possible to take some existing anatomy ontology, and through some series of operations create the required finance ontology. However, if no remnants of the anatomy ontology are preserved in the finance ontology then we would not really want to consider such behaviour to be reuse. In fact, in this case we may as well have developed the finance ontology from scratch. While the intuition of manipulating an existing ontology is certainly a necessary part of reuse, it is not sufficient to define reuse. In order to define reuse, we first consider the notion of reusability in more detail and produce a formal definition. Following this, we define the different possible reuse operations by which an ontology may be manipulated. The combination of the classification of operations and the condition of reusability will form the
5. Reusability Defined

In ontology design, we are attempting to create a set of axioms that captures some intended semantics of a set of concepts, (as specified by the requirements, likely with some application in mind). The semantics captured by a set of axioms is more than the sentences themselves, but anything that can be deduced from these sentences. A sentence may be deduced from a theory if and only if it satisfies all of the models of the theory, therefore one means of capturing this required semantics for an ontology is with the notion of intended models as described in Chapter 3. A perspective on the goal of ontology design then, is the aim of developing a set of axioms that captures these intended models, as illustrated in Figure 5.1 (originally from [42]).

It follows that to satisfy the requirements with existing theories, we should be reusing theories that in some way characterize one or more of the different domains the comprise the class of the intended models (recall, we denote these as $\mathcal{M}_{\text{intended}}$). Observe that a key distinction between reuse and traditional development is that with the concept of reuse there is an implicit constraint on the acceptable (re-)design of the axioms. Simply put, if we claim that some ontology(s) has been reused, we expect and in fact should require that some remnants of the original theory remain.

The rationale of this stipulation is that if no traces of the theory’s semantics can be found in the re-designed ontology, then the developer has essentially created an entirely new theory which may as well have been from scratch since no elements, except possibly the terms in the signature, from the original theory(s) were preserved. For any ontology $T$ that is reused to satisfy $\mathcal{M}_{\text{intended}}$, the models of $T$ must characterize at least some of the models of some part of $\mathcal{M}_{\text{intended}}$. We can restate this condition as saying that each model in $\mathcal{M}_{\text{intended}}$ must map to a model of $T$, or a model of a subtheory of $T$. Depending on the nature of the theory to be reused, $\mathcal{M}_{\text{intended}}$ may only map to some of the models of $T$ (recall, we denote the models of a theory $T$ as $\text{Mod}(T)$) as shown in Figure 5.5 (i.e. if $T$ is weaker than the required theory). On the other hand, $\mathcal{M}_{\text{intended}}$ may map to models beyond $\text{Mod}(T)$, i.e. models of some subtheory of $T$; this may occur if $T$ is either stronger or incomparable to the required theory, as shown in Figure 5.3 and Figure 5.4 respectively.
Figure 5.1: The goal of design is to have the models of the axiomatization \((O_k)\) capture the intended models \((I_k(L))\) completely, without any unintended or omitted models.

An additional factor to account for is that the class of intended models may characterize one or many different domains. For example, consider the design of an enterprise ontology: \(M_{\text{intended}}\) will likely cover concepts of organizations, actors, dates, and so on. However, we do not necessarily expect to be able to reuse a single theory that completely covers these enterprise concepts. More likely, we hope to find useful theories that contribute to the various required domains; we may reuse some theory of time, another theory of dates, and perhaps another ontology of organizations. In such cases, it is not all of \(M_{\text{intended}}\) that maps in some way to \(T\), but a reduct of \(M_{\text{intended}}\) where the models are restricted so some sub-signature (e.g. only the time-related concepts). Similarly, there are cases where \(T\) may have a larger signature than that of the intended models. For example, in the design of our enterprise ontology we may reuse a theory of time that is in fact part of a larger ontology for scheduling. In this case, the reduct of \(M_{\text{intended}}\) maps to a reduct of the models of \(T\). Therefore the diagrams that we have been considering are not necessarily capturing mappings between the intended models and the models of the reused theory, but the reducts of these models as shown more precisely in Figure 5.5.

To account for mismatches in the scope of the required and reused ontology’s concepts, we consider mappings between the reducts of the models. Thus \(L_1\) may in fact be a sub-language (sub-signature) of \(\sigma(M_{\text{intended}})\), and similarly \(L_2\) may be a sub-language (sub-signature) of the candidate’s signature \(\sigma(T)\). \(^1\) Here, we extend the usual meaning of a reduct of a single model to some sub-signature of its original signature to apply to an entire

\(^1\)Recall, we denote the signature of a given theory \(T\) or class of models \(M\) by \(\sigma(T), \sigma(M)\), respectively.
class of models. Formally, we denote this $\text{Red}(\mathcal{M}, \sigma)$, and define it as follows:

**Definition 11.** A reduct of a class of models $\mathcal{M}$ to some signature $\sigma$ is defined as the class structures consisting of the reducts of each model $\mathcal{M}$ in $\mathcal{M}$ to $\sigma$.

$$\text{Red}(\mathcal{M}, \sigma) = \{ N : N \cong M \mid \sigma, M \in \mathcal{M} \}$$

Collecting all of these observations, we can refine our intuitions to say more specifically that if a theory $T$ is reusable to satisfy $\mathcal{M}^{\text{intended}}$, then there must be a mapping from some reduct of $\mathcal{M}^{\text{intended}}$ to some reduct of the models of $T'$, a subtheory of $T$ with the same signature. Formally,

**Definition 12.** $T$ is reusable for $\mathcal{M}^{\text{intended}}$ iff:

there is a mapping $\pi$ from some reduct of $\mathcal{M}^{\text{intended}}$ to a reduct of $\text{Mod}(T')$.

$$\pi : \text{Red}(\mathcal{M}^{\text{intended}}, \sigma_1) \rightarrow \text{Red}(\text{Mod}(T'), \sigma_2)$$

where:

- $T' \leq T$ (i.e. $T$ is a nonconservative extension of $T'$, and $T$ and $T'$ are in the same hierarchy, as defined in the previous chapter.)
- $\sigma_1 \subseteq \sigma(\mathcal{M}^{\text{intended}})$
- $\sigma_2 \subseteq \sigma(T')$

It is straightforward to extend this definition to apply to a collection of ontologies. The same intuition applies:

**Definition 13.** For a set of ontologies, $T_1, \ldots, T_n$, we say that the set $T_1, \ldots, T_n$ is reusable for $\mathcal{M}^{\text{intended}}$ iff each $T_i$ is reusable for $\mathcal{M}^{\text{intended}}$. 
Figure 5.3: Intuitively, if a theory is reusable for some required ontology, we expect to find some part of it in the resulting ontology. If the theory is stronger than required, the intended models will map to models of a subtheory its axiomatization.

This notion of reusability can be captured similarly, from the perspective of the theories’ axiomatizations. Note that owing to our definition of a theory being the logical closure of a set of axioms, the $\subseteq$ symbol denotes a subtheory as opposed to simply a subset of axioms. We use $Th(\mathcal{M})$ to denote the axiomatization of a class of models $\mathcal{M}$. 

**Theorem 4.** A set of ontologies $T_1, \ldots T_n$ is reusable for $\mathcal{M}^{\text{intended}}$ iff $Th(\mathcal{M}^{\text{intended}})$ contains non-trivial subtheories of $T_1, \ldots, T_n$.

**Proof.** $\Rightarrow$

By Definition [13] if the set of ontologies $T_1, \ldots T_n$ is reusable for $\mathcal{M}^{\text{intended}}$, then for each $T_i$, $i = 1, \ldots, n$, we must have:

- some $R_i = Red(\mathcal{M}^{\text{intended}}, \sigma_1)$ where $\sigma_1 \subseteq \sigma(\mathcal{M}^{\text{intended}})$, and
- some $\mathcal{N}_i = Red(Mod(T'_i), \sigma_2)$ where $T'_i \leq T_i$ and $\sigma_2 \subseteq \sigma(T_i)$
- and there must be some mapping $\pi_i$, s.t. $\pi_i : R_i \rightarrow \mathcal{N}_i$

Since we are assuming that the mapping does not alter the signature of the models (in this simplified presentation we assume that no logic- or signature-translations are required), observe that we must have:

$Th(R_i) \models Th(\mathcal{N}_i)$

$\Rightarrow Th(\mathcal{N}_i) \subseteq Th(R_i)$

By definition of a reduct, the axiomatization of a reduct of some class of models $Th(Red(\mathcal{M}, \sigma))$ must be a subtheory of the axiomatization of that class of models $Th(\mathcal{M})$, therefore:

$Th(\mathcal{N}_i) \subseteq T'_i \subseteq T_i$ and
CHAPTER 5. ATTAINING THE BENEFITS OF REUSE

5.2 Definition of Reuse

While it is tempting to interpret Theorem 4 as a definition of reuse, it would be inaccurate to do so. Certainly, for any ontology(s) that has been reused, we expect that it must have been reusable and thus we expect the results of reuse are captured by Theorem 4. In fact, the theorem captures the basic intuition of reuse that motivated the definition of reusability: in order for Design to qualify as reuse, we expect some remnants of the original theory(s) to remain. However, reusability is a necessary but not sufficient condition for reuse. There is an extralogical condition that must be accounted for in order to completely capture what it means to reuse an ontology. Unlike reusability, reuse is not a static property between theories; reuse refers to an act that is performed with some
Figure 5.5: Intuitively, if a theory is reusable for some required ontology, we expect to find some part of it in the resulting ontology.

existing theories, in the design of an ontology.

To make this distinction more concrete, consider the Process Specification Language (PSL) ontology [32]. PSL was developed from-scratch and contains, among other things, an axiomatization of temporal concepts. While it was later shown that PSL’s theory of time could be mapped to existing time ontologies in COLORE, none of these theories was actually reused to design PSL. Similarly, consider a scenario in which we have some ontology $T_{onto}$ which was designed, via the reuse of some $T_1$. There may be some other ontologies, $T_2, T_3$ that were also reusable for the design of $T_{onto}$, however the designer elected to reuse $T_1$. The design work involved to reuse $T_1$ as opposed to $T_2$ or $T_3$ will be distinct, and the reused theory may even have some influence on the final form of $T_{onto}$ (as we have mentioned previously, in practice $\mathfrak{M}^{intended}$ is not completely known thus there may be some flexibility during development). It is important that this distinction is reflected in order to correctly capture the definition of reuse.

Informally, reuse is the act of applying some operation(s) to a given, reusable theory(s), such that the final result axiomatizes the intended models. To formally define reuse, these operations must be completely identified and defined. As discussed in Chapter 2, many approaches to reuse, such as ontology fusion, merging, and extension, have been identified in varying degrees of detail in the current state. However, these approaches have not been defined with respect to a complete definition of reuse, and in some cases they have not been defined formally at all. In this section, we provide a precise definition for a set of operations that completely covers the act of reuse. The terms used here should be interpreted independently of those that have been identified/defined in the literature. No relationships should be inferred or assumed due to a similarity of terms or descriptions. In Section 5.2.1 we discuss the relationships between this classification of reuse types and existing, prevalent types of reuse.

The following are the 4 distinct reuse operations by which an ontology may be manipulated for reuse (excluding translations between signatures and logics, as per the assumption made earlier in this chapter): as is, extraction, extension, and combination.
As is refers to a sort of null operation. This corresponds to the reuse of an ontology directly in the plug-and-play paradigm with no modifications of any sort. In this sense, it is analogous to an identity function and we consider it to be a trivial operation. It can be formally defined as follows:

**Definition 14.** \( \text{as\_is}(T) = T \)

**Extraction** refers to the reuse of an ontology via a removal of some of the original axioms. It can be formally defined as follows:

**Definition 15.** \( \text{extraction}(T, T^-) = T / T^- \)

Where the / symbol denotes the difference between two theories.

The first two operations addressed the sort of do-nothing operation, and the operation to remove axioms from a theory. Now we consider the operations to add axioms to a theory; the way that this occurs depends on the source of the axioms – they could be new axioms, created during design by the ontology developer, or they could be existing axioms, reused from some other ontology. It is important to make this distinction because these different operations indicate a very different design process and workload. If the axioms were reused from another ontology, the design work is minimal, however if they are new axioms then the developer must have invested some time to design them from-scratch. The distinction between these two types of axiom addition also has potential implications for the benefit of shareability. If the additional axioms were reused from some other ontology it may indicate that shareability will also be supported with this ontology: at the very least it indicates that shareability is something that should be considered, and addressed in the metadata. We formalize this distinction by considering whether the additional axioms may be found in some repository. If axioms are added to an ontology by reuse (addition of) axioms from another ontology, then a combination has occurred. Otherwise, the addition is simply an extension of the ontology with additional axioms. We make reference to a single repository for simplicity, however it is straightforward to see that the definition and subsequent results also apply for any number of repositories.

**Extension** refers to the reuse of an ontology via the creation and introduction of new axioms. It can be formally defined as follows:

**Definition 16.** Let \( S \) be some ontology repository. \( \text{extension}(T, T^+) = T \cup T^+ \) where \( T^+ \notin S \)

**Combination** refers to the reuse of a theory via the addition of (“in combination with”) other reused ontology(s). It can be formally defined as follows:

**Definition 17.** Let \( S \) be some ontology repository. \( \text{combination}(T_1, T_2) = T_1 \cup T_2 \) where \( T_2 \in S \)

While it is intuitive to consider reuse as the application a sequence of these operations, so long as a theory or set of theories \( T_1, ... T_n \) is reusable for \( \text{Th}(\mathbb{M}_{\text{intended}}) \), the order in which reuse operations are applied does not affect the final result. The order may be of some importance in maintaining consistency throughout the series of operations, however this is not a requirement of design.

**Theorem 5.** The operations for reuse are commutative.

**Proof.** To illustrate this, we consider each pairing of reuse operations and show that either order of application produces the same results. It is straightforward to see that this then extends to any series of operation applications.
**As is and Extraction** Consider the application of as.is to some $T$, followed by some extraction.

By definition, $\text{as.is}(T) = T$, thus this can be expressed simply as: $T/T^-$. Similarly, the application of some extraction followed by as.is reuse can be expressed as: $T/T^-$. Therefore, to prove that as.is and extraction are commutative operations we must show that $T/T^- = T/T^-

The result is trivial.

**As is and Extension** Similar to above, from the definition of as.is the result is trivial.

**As is and Combination** Similar to above, from the definition of as.is the result is trivial.

**Extraction and Extension** Consider the application of extraction to some $T$, followed by some extension.

This can be expressed as: $T/T^- \cup T^+$. Similarly, some extension of $T$ followed by an extraction can be expressed as: $T \cup T^+ / T^-$. Therefore, to prove that extraction and extension are commutative operations we must show that $T/T^- \cup T^+ = T \cup T^+ / T^-

Consider $T/T^-$. It is straightforward to see that the removal of any subtheory of $T$ (i.e. an extraction) is a subtheory of $T$: $T/T^- \subset T$

Now, say we add some theory to the extraction of $T$ and to $T/T^-$. It is straightforward to see that the the extraction, with some addition, will also be a subtheory of $T$, with the same addition: $T \cup T^+ / T^- \supset T \cup T^+$

Likewise, any subtheory that we remove from both sides cannot affect this relationship; at most we can reduce $T \cup T^+$ such that it is equivalent to $T/T^- \cup T$, we cannot possibly to reverse the subsumption relationship through extraction. Therefore, if perform an extraction of $T^-$ again on both sides we result with: $T/T^- \cup T^+ \subseteq T \cup T^+ / T^-

Now, consider $T \cup T^+$. It is straightforward to see that any extension of $T$ subsumes $T$ itself: $T \cup T^+ \supset T$

This subsumption relationship will be maintained, with the possibility of equivalence, should any subtheory be removed from both sides: $T \cup T^+ / T^- \supset T/T^-$

Again, any subtheory added to both sides will maintain the subsumption relationship will be maintained.

Adding $T^+$ yields: $T \cup T^+ / T^- \supset T/T^- \cup T^+

Since we have shown: $T \cup T^+ / T^- \supset T/T^- \cup T^+$

and $T/T^- \cup T^+ \subseteq T \cup T^+ / T^-

Then we must have: $T/T^- \cup T^+ = T \cup T^+ / T^-$

**Extract and Combine** The result is the same as above, except instead of $T^+$, we have $T_2$ representing any possible theory that will be combined with $T$.

**Extend and Combine** Trivial to see that: $T \cup T^+ \cup T_2 = T \cup T_2^+ = T^+$

Observe that there are not only different possible sequences but in fact many different sets of operations that achieve the same result. Such sets may be differentiated through redundant or unnecessary operations. For example, consider repeated applications of extension and extraction operations that add and remove the same axioms. This may seem bizarre, but consider a design scenario where one approach is attempted, only to find
it does not have the desired effect and it is therefore undone or modified in a subsequent step. It is useful to distinguish those operations that are necessary to achieve the reuse of a particular theory(s) for some $M^{\text{intended}}$, and those that are simply allowed by the definition of reuse.

A necessary operation for reuse is defined with respect to a particular set of operations applied to $T_1, ..., T_n$ to axiomatize $M^{\text{intended}}$.

**Definition 18.** Let $\mathcal{X}$ be a set of reuse operations applied to some ontology(s) $T_1, ..., T_n$ to axiomatize $M^{\text{intended}}$. A non-trivial reuse operation $x$ is a necessary operation for the reuse of ontology(s) $T_1, ..., T_n$ to axiomatize $M^{\text{intended}}$ iff any subset of the set of operations applied to $T_1, ..., T_n$ to axiomatize $M^{\text{intended}}$ contains $x$.

$x \in \mathcal{X}'$ for all $\mathcal{X}' \subset \mathcal{X}$, where $\mathcal{X}'$ applied to $T_1, ..., T_n$ axiomatises $M^{\text{intended}}$.

Again, consider the definition of reuse – this time based on the reuse operations we’ve just defined. Such a definition appeals to our intuition: reuse is the application of some set of reuse operations; however the act of reuse cannot be defined via these operations alone. It is not the case that any set of these operations corresponds to an act of reuse. For example, consider a scenario in which we extract a subtheory from some existing ontology, extend it with some new axioms, and then perhaps extract a subtheory again, such that no remnants of the original theory remain. This violates a key condition for reuse that is captured in the definition for reusability. We cannot have such possibilities included within the definition of reuse. To completely define what we mean by reuse, we must incorporate both the logical conditions identified previously and the extralogical conditions of the operations performed.

Reuse is defined as the act of performing some set of operations on some existing ontology(s) $T_1, ..., T_n$ that is reusable for some $M^{\text{intended}}$, in order to transform $T_1, ..., T_n$ to $Th(M^{\text{intended}})$. Formally,

**Definition 19.** $T_1, ..., T_n$ are reused for $M^{\text{intended}}$ iff

- $T_1, ..., T_n$ are reusable for $M^{\text{intended}}$, and
- some set of reuse operations applied to $T_1, ..., T_n$ axiomatizes $M^{\text{intended}}$

Observe that reusability is a necessary condition for reuse. Similarly, the defined operations are necessary in an indirect way – if reuse has occurred, then some sequence of the reuse operations must have been applied to the ontologies to result in an axiomatization of the intended models. On the other hand, both the conditions of reusability and the application of reuse operations are required to form the sufficient condition for reuse.

Admittedly the reuse of some $T_1, ..., T_n$ for $M^{\text{intended}}$ may not occur in a particularly structured fashion. Perhaps none of the operations were consciously identified or performed, however if the ontologies were reused then it must be possible to describe the design that took place via the defined operations. Note that for the remainder of the Chapter, in the results that follow we assume that both the ontology(s) being reused, $T$, and $Th(M^{\text{intended}})$ are axiomatized only in their primitive signature; in other words we exclude all conservative definitions as they complicate the presentation of this material while having no impact on the underlying logic.

**Theorem 6.** Let $\mathcal{S}$ be some ontology repository, and let $T_1, ..., T_n$ be some ontologies in $\mathcal{S}$. If $T_1, ..., T_n$ are reusable for $M^{\text{intended}}$, then there exists some set of reuse operations that can be applied to $T_1, ..., T_n$ to axiomatize $M^{\text{intended}}$.

**Proof.** Assume the theorem is false. Then we must have some $T_1, ..., T_n$ reused for $M^{\text{intended}}$ that cannot be axiomatized with some sequence of the reuse operations.

There are 2 possible cases: reuse of a single theory or reuse of multiple theories.
Case 1: reuse of a single theory \( n = 1 \) i.e. a single theory \( T \) is reused for \( Th(\mathfrak{M}_{\text{intended}}) \).

By definition, if \( T \) is reused for \( Th(\mathfrak{M}_{\text{intended}}) \) then \( T \) is reusable for \( \mathfrak{M}_{\text{intended}} \) and by Theorem 4:

There exists a \( T' \) s.t. \( T' \subseteq T \) and \( T' \subseteq Th(\mathfrak{M}_{\text{intended}}) \).

It is straightforward to see that in this case one of the following 4 situations must be true:

1. \( T' \subseteq T \) and \( T' \subseteq Th(\mathfrak{M}_{\text{intended}}) \)
   
   There exists a \( T^- \) such that \( \text{extraction}(T, T^-) = T' \)
   
   \( T \) reused for \( T' \) via \text{extraction} of some \( T^- \)
   
   \( \rightarrow T' \subseteq Th(\mathfrak{M}_{\text{intended}}) \)
   
   There exists a \( T^+ \) such that \( \text{extension}(T', T^+) = Th(\mathfrak{M}_{\text{intended}}) \)
   
   \( T' \) reused for \( Th(\mathfrak{M}_{\text{intended}}) \) via \text{extension} with some \( T^+ \).
   
   \( \text{extension}(\text{extraction}(T, T^-), T^+) = Th(\mathfrak{M}_{\text{intended}}) \)

   Therefore the assumption cannot hold in this situation as it corresponds to reuse of \( T \) via extraction of some \( T^- \) and extension with some \( T^+ \) to axiomatize \( Th(\mathfrak{M}_{\text{intended}}) \). It is clear from the definitions of the reuse operations that \text{extraction} and \text{extension} are the only possible operations to axiomatize \( Th(\mathfrak{M}_{\text{intended}}) \).

2. \( T' \subseteq T \) and \( T' = Th(\mathfrak{M}_{\text{intended}}) \)
   
   There exists a \( T^- \) such that \( \text{extraction}(T', T^-) = Th(\mathfrak{M}_{\text{intended}}) \)

   The assumption cannot hold in this situation as it corresponds to reuse of \( T \) via extraction of some \( T^- \). It is clear from the definitions of the reuse operations that \text{extraction} is the only possible operation to axiomatize \( Th(\mathfrak{M}_{\text{intended}}) \).

3. \( T' = T \) and \( T' \subseteq Th(\mathfrak{M}_{\text{intended}}) \)
   
   There exists a \( T^+ \) such that \( \text{extension}(T', T^+) = Th(\mathfrak{M}_{\text{intended}}) \)

   This cannot hold as it corresponds to reuse of \( T \) via extension with some \( T^+ \). It is clear from the definitions of the reuse operations that \text{extension} is the only possible operation to axiomatize \( Th(\mathfrak{M}_{\text{intended}}) \).

4. \( T' = T \) and \( T' = Th(\mathfrak{M}_{\text{intended}}) \)
   
   \( \text{as.is}(T) = Th(\mathfrak{M}_{\text{intended}}) \)

   This cannot hold as it corresponds to \text{as.is} reuse of \( T \) for \( Th(\mathfrak{M}_{\text{intended}}) \). It is clear from the definitions of the reuse operations that \text{as.is} is the only possible operation to axiomatize \( Th(\mathfrak{M}_{\text{intended}}) \).

Therefore when \( n = 1 \) the assumption cannot hold so we can conclude that any reuse of a single theory can be defined by some sequence of reuse types.

Case 2: reuse of multiple theories \( n > 1 \)

In order to reuse multiple theories, the \textit{combination} operation must be applied at some point, to collect all of the theories \( T_1, \ldots, T_n \) together (i.e. \( n - 1 \) times). We consider this necessary operation first, (recall that the operations are commutative) and then show how the result reduces to Case 1, for which we have shown the assumption cannot hold.

Again, by Theorem 4:

There exists a \( T'_1, \ldots, T'_n \) s.t. \( T'_1 \subseteq T_1, \ldots, T'_n \subseteq T_n \) and \( T'_1 \cup \ldots \cup T'_n \subseteq Th(\mathfrak{M}_{\text{intended}}) \)

Consider all \( n \) theories collectively, let

\( T_{\text{new}} = T_1 \cup \ldots \cup T_n = \text{combination}(T_1, \text{combination}(\ldots, \text{combination}(T_{n-1}, T_n))) \)

i.e. by definition, \( T_1, \ldots, T_n \) are reused for \( T_{\text{new}} \) via \text{combination}.

Let \( T'_{\text{new}} = T'_1 \cup \ldots \cup T'_n \)
\[ T'_{\text{new}} \subseteq T_{\text{new}} \]
\[ T'_{\text{new}} \subseteq \mathcal{M}(\text{intended}) \]

By definition, \( T'_{\text{new}} \) is reused for \( \mathcal{M}(\text{intended}) \) and we now have \( n = 1 \) for which we’ve shown (Case 1) the assumption is impossible (i.e. a subsequent sequence of operations exists for any scenario). Therefore we can conclude that any reuse of a set of \( n \) theories where \( n > 1 \) can also be defined by some sequence of reuse types.

### 5.2.1 Other Approaches to Reuse

In Theorem 6, we showed that any ontologies reusable for some \( \mathcal{M}(\text{intended}) \) can be reused to axiomatize \( \mathcal{M}(\text{intended}) \) via some set of the reuse operations defined here. While there are other approaches to reuse that are not directly captured here, it follows that any of these previously defined approaches to reuse are also expressible as some combination of the operations defined here. As an example, we show how the task of a merge - a popular paradigm for reuse - may be decomposed into a set of the reuse operations defined here.

The term merge most often describes an approach to reuse in which two (or more) existing ontologies describing different domains are reused to create a new ontology that requires the combination of such concepts. For example, we might reuse an ontology of dates and an ontology of processes (among others) to create an ontology for some e-commerce application. Typically, the act of a merge is thought of as not only the combination of the required ontologies, but also the addition of ‘bridge’ axioms that combine the domains in a meaningful way.

Formally then, we define the merge of a set of ontologies \( T_1, ..., T_n \) as: \( T_1 \cup ... \cup T_n \cup T_{\text{bridge}} \) where the signature of the bridge axioms contains concepts from at least subsets of the original ontologies, and possibly extends beyond them. Where \( T'_1 \subseteq T_1, ..., T'_n \subseteq T_n \), we have \( \sigma(T'_1 \cup ... \cup T'_n) \subseteq \sigma(T_{\text{bridge}}) \), as \( T_{\text{bridge}} \) not only combines concepts from the set of ontologies, but may employ new concepts in order to express the relationships of interest in the combination. It is straightforward to see that the merge of a set of ontologies \( T_1, ..., T_n \) for some \( \mathcal{M}(\text{intended}) \) can be described as a combination of \( T_1, ..., T_n \), followed by an extension with the bridge axioms. Formally:
\[
\text{extension}\left(\text{combination}(T_1, \text{combination}(..., \text{combination}(T_{n-1}, T_n))), T_{\text{bridge}}\right).
\]

It should be noted that instances of reuse by what is sometimes referred to as inspiration [65] will likely be excluded from the definition of reuse presented here; this is an intentional restriction. While inspiration may be a valid technique for ontology developers in certain situations we find no motivation to include it in our definition or classifications. Even from a documentation perspective, it is beneficial to distinguish actual reuse of an ontology as defined here, from inspiration drawn from an ontology. They are distinct in both their execution and their results. Drawing inspiration from an existing ontology does not produce any of the supposed outcomes or benefits of reuse; even the reduction in work affects only part of the design stage (i.e. doesn’t benefit the actual encoding of the axioms). At the core of this exclusion is the fact that reuse refers to a true, concrete use of the ontology and its content, whereas inspiration in fact ‘uses’ an ontology in a very different, less formal and less substantial way.

### 5.3 Reduction of Design Work

The first of the two benefits of reuse we will tackle is the supposed reduction of design work. We observe that this benefit follows from the definition of reusability: so long as a theory \( T \) is reusable for \( R \), then by Theorem 4 some of \( T \) (i.e. a subtheory) will be extended by \( R \), and therefore the effort that would have been required to design that subtheory of \( T \) in \( R \) may be saved. Therefore the definition of reusability has already made the benefit
of Reduced Design Work more tractable. By selecting candidates that are reusable according to our definition, this benefit is guaranteed to result. Beyond this, it would be useful to understand this resulting benefit at a finer level of granularity. A natural question is – to what degree is the design work reduced? To better understand this, we should also consider precisely what design work will be required. This information is also useful in that it provides guidance for the reuse of a given theory(s), consequently further reducing the work required to identify the necessary reuse operations for Design.

In this section we identify how the question of what design work is required may be answered. We find that the definition of reuse may be leveraged to further reduce design work by providing guidance for the reuse of a given set of reusable theories \(T_1,\ldots,T_n\) through the defined operations. In fact, the proof of Theorem 6 indicates that not only does there always exist set of necessary operations for the reuse of any given \(T_1,\ldots,T_n\) to axiomatize some \(\mathcal{M}_{\text{intended}}\) that they are reusable for, but that it is possible to explicitly identify such a set. The set of reuse operations that are necessary to axiomatize the intended models, can be leveraged to guide the developer during the design, thereby simplifying the task and reducing the work required.

Here, we extend the results of the proof for Theorem 6 to show how given a set of theories \(T_1,\ldots,T_n\) that are reusable for \(\mathcal{M}_{\text{intended}}\), we can explicitly identify what reuse operations are necessary to axiomatize \(\mathcal{M}_{\text{intended}}\). First, consider the reuse of a single theory \(T\) for \(\mathcal{M}_{\text{intended}}\). Rather than consider the relationship between the subset of \(T\) that is preserved in \(Th(\mathcal{M}_{\text{intended}})\), we consider the relationship between \(T\) and \(Th(\mathcal{M}_{\text{intended}})\) directly. In practice, \(Th(\mathcal{M}_{\text{intended}})\) will likely not be known completely and thus some approximation (such as the set of CQs, as discussed in previous chapters) will be substituted.

There are 4 possible relationships between \(T\) and \(Th(\mathcal{M}_{\text{intended}})\). For each relationship, the necessary reuse operation to axiomatize \(Th(\mathcal{M}_{\text{intended}})\) follows by definition of the operations:

- \(T \models Th(\mathcal{M}_{\text{intended}}) \rightarrow \text{extraction}(T, T^-) = Th(\mathcal{M}_{\text{intended}})\) for some \(T^-\)
- \(Th(\mathcal{M}_{\text{intended}}) \models T \rightarrow \text{extension}(T, T^+) = Th(\mathcal{M}_{\text{intended}})\) for some \(T^+\)
- \(T = Th(\mathcal{M}_{\text{intended}}) \rightarrow \text{as_is}(T)\)
- \(T \parallel Th(\mathcal{M}_{\text{intended}}) \rightarrow \text{extension(extraction}(T, T^-), T^+)\) for some \(T^-, T^+\) where the \(\parallel\) symbol denotes that two theories are incomparable

In order to account for the reuse of multiple theories \(T_1,\ldots,T_n\) (Case 2 in the proof of Theorem 6) we apply the combination operation in a similar fashion. However, in this case since we are assessing the entailment relationship between theories, we must take care to avoid any inconsistencies. We first consider the reuse of each \(T_1,\ldots,T_n\) from the single ontology perspective, as shown above, and apply extraction operation where it is indicated. This is done to avoid the potential inconsistencies that may result from the application of the combination operation. Since we have shown that the reuse operations are commutative, this has no bearing on the result. What it does mean is that for multiple theories, the guidance for reuse operations cannot be known completely a priori. After the extraction operation has been applied to each individual theory where required, the combination operation may be applied to collect all of the theories. We can then consider the entailment relationship between combination\(\left(T_1, (\ldots, \text{combination}(T_{n-1}, T_n))\right)\) and \(Th(\mathcal{M}_{\text{intended}})\) in order to identify any remaining operations.

Therefore, while not possible completely a priori we have illustrated how with the results of Theorem 6 it is possible to determine the necessary reuse operations for any \(T_1,\ldots,T_n\) reusable for \(Th(\mathcal{M}_{\text{intended}})\). Not only is it a straightforward consequence of the definition of reusability that some Design effort be reduced, but the necessary
reuse operations can be identified to provide explicit guidance for the reuse of any (reuseable) theory, thus further reducing the effort required for reuse.

5.4 Interoperability

The second main benefit of ontology reuse is the promise of interoperability. The idea is that if a developer reuses a theory, shareability will naturally result; thus they can expect to immediately have interoperability with the theory they have reused, as well as any other ontologies that have reused that theory. The provision of this benefit hinges on the preservation of the reused theory’s semantics. The shareability and resulting interoperability is ‘free’ so long as the semantics are preserved during reuse. The difficulty in the current state is that this preservation of semantics is not guaranteed by the definition of reuse, nor is it the norm in ontology development.

One approach to ensuring this benefit is fulfilled would be to revise and restrict the definition of reuse, such that we can guarantee the preservation of semantics. However it is unrealistic to expect the adoption of such a restrictive practice, and any attempt to somehow enforce such a decision would risk deterring reuse altogether. Instead, we aim to provide the developer with the tools to better understand reuse, and how decisions that are made in the reuse process impact the potential benefit of interoperability. The result of Theorem 5 can be extended to make the benefit of shareability tractable. The identification of necessary reuse operations can be interpreted to determine whether the semantics of any reusable theory will be preserved in the axiomatization of $M_{intended}$.

In order to determine whether the semantics of a reused theory are preserved, we need to identify the reuse operations at a more detailed level. In the previous section, we showed that for any reusable theory, its relationship with $Th(M_{intended})$ could be assessed in order to determine the reuse operations necessary to axiomatize $M_{intended}$. To assess the impact on semantics, we need each reuse operation to be defined so as to indicate the logical relationship between its result and the theory it is applied to. For example, is the semantics of $T$ preserved when it is extended with some $T^+$? This level of detail will make it possible for the developer to determine what, if any, benefits of interoperability they can expect to achieve for a given candidate or set of candidates. This can be done a priori before the task of reuse is performed, therefore the implications of the developer’s choice of ontology to reuse will be completely transparent. Further, this information may be applied post-development as ontology metadata in order to inform potential users of what sort of interoperability they may expect.

5.4.1 Specialized Reuse Operations

Extraction, extension and combination may be performed in several distinct ways, each of which has a different intuition and impact on the resulting ontology. In the following, we explicitly define subtypes of the reuse operations based on a more precise identification of the changes to the original theory. This allows for a more detailed analysis of the implications of any given reuse scenario. It facilitates recognition of the relationships between concepts in the original ontology(s) and the resulting ontology being developed via reuse. Most notably, this enables the identification of shareability that is attained or lost for concepts in the new ontology.

**Extraction** recall, $\text{extraction}(T, T^-) = T / T^-$ where $/$ denotes the difference between theories. We identify 3 distinct specializations of this operation, based on the relationship of the result, $T / T^-$, to the original theory $T$.

**Domain Extraction:** an entire domain (set of concepts) is completely extracted from the original ontology

**Definition 20.** $\text{domain\_extraction}(T, T^-) = T / T^- \text{ where } \sigma(T^-) \cap \sigma(T/T^-) = \emptyset$
Observe that $T$ conservatively extends $\text{domain} \_ \text{extraction}(T/T^-)$.

**Weakening Extraction**: the semantics of the original ontology are weakened by the operation while its scope remains the same.

**Definition 21.** $\text{weakening} \_ \text{extraction}(T, T^-) = T/T^- \text{ where } \sigma(T/T^-) = \sigma(T)$

Observe that $T$ non-conservatively extends $\text{weakening} \_ \text{extraction}(T, T^-)$.

**Weakening Domain Extraction**: the semantics of the original ontology are weakened and only some of the concepts are reused with this operation. No part of this extraction may be comprised of a $\text{weakening} \_ \text{extraction}$ or a $\text{domain} \_ \text{extraction}$.

**Definition 22.** $\text{weakening} \_ \text{domain} \_ \text{extraction}(T, T^-) = T/T^- \text{ where } \sigma(T/T^-) \subset \sigma(T)$ and there does not exist a $T_{\text{sub}}^- \subset T^-$ such that $
\sigma(T_{\text{sub}}^-) \cap \sigma(T/T_{\text{sub}}^-) = \emptyset \text{ or } \sigma(T/T_{\text{sub}}^-) = \sigma(T)$

Observe that $T$ non-conservatively extends $\text{weakening} \_ \text{domain} \_ \text{extraction}(T, T^-)$.

It is straightforward to see that domain extraction, weakening extraction, and weakening domain extraction are the only ways in which reuse via extraction can occur.

**Extension**  
Recall, some repository $S$, $\text{extension}(T, T^+) = T \cup T^+$ where $T^+ \notin S$. We identify 3 distinct specializations of this operation, based on the relationship of the axioms added, $T^+$, to the original ontology, $T$

**Domain Extension**: the original ontology is extended via a new set of axioms created by the developer, $T^+$, in a completely new (distinct) domain.

**Definition 23.** Let $S$ be some ontology repository. $\text{domain} \_ \text{extension}(T, T^+) = T \cup T^+$ where $T^+ \notin S$ and where $\sigma(T) \cap \sigma(T^+) = \emptyset$

Observe that $\text{domain} \_ \text{extension}(T, T^+)$ conservatively extends $T$.

**Strengthening Extension**: the original ontology is extended via a new set of axioms created by the developer, $T^+$, such that its semantics are strengthened while maintaining its scope.

**Definition 24.** Let $S$ be some ontology repository. $\text{strengthening} \_ \text{extension}(T, T^+) = T \cup T^+$ where $T^+ \notin S$

and where $\sigma(T^+) \subseteq \sigma(T)$

Observe that $\text{strengthening} \_ \text{extension}(T, T^+)$ non-conservatively extends $T$.

**Strengthening Domain Extension**: the original ontology is extended via a new set of axioms created by the developer, $T^+$, both strengthening its original concepts and adding to them, thereby expanding the scope of its domain. No part of this extension may be comprised of a $\text{domain} \_ \text{extension}$ or a $\text{strengthening} \_ \text{extension}$.

**Definition 25.** Let $S$ be some ontology repository. $\text{strengthening} \_ \text{domain} \_ \text{extension}(T, T^+) = T \cup T^+$ where $T^+ \notin S$

and where $\sigma(T) \subset \sigma(T^+)$, or

$\sigma(T) \cap \sigma(T^+) \neq \emptyset$, $\sigma(T) \notin \sigma(T^+)$, $\sigma(T) \nsubseteq \sigma(T^+)$ (i.e. signatures overlap)
and there does not exist a $T'_{sub} \subset T^+$ such that:
\[
\sigma(T'_{sub}) \subseteq \sigma(T) \quad \text{or} \quad \\
\sigma(T'_{sub}) \cap \sigma(T) = \emptyset
\]

Observe that $\text{strengthening-domain-extension}(T, T^+) \nonconservatively$ extends $T$.

It is straightforward to see that domain extension, strengthening extension, and strengthening domain extension are the only ways in which reuse via extension can occur.

**Combination**  Recall that the combination operation is similar to the extension operation, but represents the case where the extension comprises axioms reused from another ontology. Let $\mathcal{S}$ be some ontology repository, the combination operation is defined: $\text{combination}(T, T_2) = T \cup T_2$ where $T_2 \in \mathcal{S}$. We identify 3 distinct specializations of this operation, based on the relationship between the theory being combined, $T_2$ and the original ontology $T$.

**Domain Combination:** The original ontology, $T$ is combined with another ontology $T_2$ that defines a completely distinct domain.

**Definition 26.** Let $\mathcal{S}$ be some ontology repository. $\text{domain-combination}(T, T_2) = T \cup T_2$ where $T_2 \in \mathcal{S}$ and where $\sigma(T) \cap \sigma(T_2) = \emptyset$

Observe that $\text{domain-combination}(T, T_2)$ conservatively extends $T$ (and $T_2$).

**Strengthening Combination:** The original ontology, $T$, is combined with another ontology that defines the same domain, $T_2$, such that their semantics are strengthened while maintaining the original scope.

**Definition 27.** Let $\mathcal{S}$ be some ontology repository. $\text{strengthening-combination}(T, T_2) = T \cup T_2$ where $T_2 \in \mathcal{S}$ and where $\sigma(T_2) \subseteq \sigma(T)$

Observe that $\text{strengthening-combination}(T, T_2)$ non-conservatively extends $T$ (and $T_2$).

**Strengthening Domain Combination:** The original ontology, $T$, is combined with another ontology $T_2$ that strengthens the concepts of $T$ while also introducing new ones. No part of this extension may be comprised of a domain_extension or a strengthening_extension.

**Definition 28.** Let $\mathcal{S}$ be some ontology repository. $\text{strengthening-domain-combination}(T, T_2) = T \cup T_2$ where $T_2 \in \mathcal{S}$ and where $\sigma(T) \subset \sigma(T_2)$, or $\sigma(T) \cap \sigma(T_2) \neq \emptyset$, $\sigma(T) \not\subseteq \sigma(T_2)$, $\sigma(T) \not\supseteq \sigma(T_2)$ (i.e. signatures overlap) and there does not exist a $T'_2 \subset T_2$ such that:
\[
\sigma(T'_2) \subseteq \sigma(T) \quad \text{or} \quad \\
\sigma(T'_2) \cap \sigma(T) = \emptyset
\]

Observe that $\text{strengthening-domain-combination}(T, T_2)$ non-conservatively extends $T$ (and $T_2$).

It is straightforward to see that domain combination, strengthening combination, and strengthening domain combination are the only ways in which reuse via combination can occur.
5.4.2 Identifying Necessary Specialized Operations for Reuse

Similar to the identification of reuse operations described in the previous section, we can assess the relationship between a set of ontologies and any set of intended models $\mathcal{M}^{\text{intended}}$ they are reusable for, to show there will always be a set of necessary specialized operations to axiomatize the intended models with the ontology(s).

**Theorem 7.** Let $\mathcal{S}$ be some ontology repository, and let $T_1, \ldots, T_n$ be some ontologies in $\mathcal{S}$. If a theory or set of theories $T_1, \ldots, T_n$ is reusable for $\mathcal{M}^{\text{intended}}$ there exists a set of necessary specialized reuse operations on $T_1, \ldots, T_n$ to axiomatize $\mathcal{M}^{\text{intended}}$.

**Proof.** Either a single theory $T$ or multiple theories $T_1, \ldots, T_n$ may be reused. We’ll consider the reuse of a single theory first.

**Reuse of a Single Theory** First, we consider the reuse of a single theory $T$ for $\mathcal{M}^{\text{intended}}$; therefore we do not need to account for the Combination operation. Assuming that $T$ is reusable for $\mathcal{M}^{\text{intended}}$, we can completely characterize all possible reuse cases with the relationship between $T$ and $\text{Th}(\mathcal{M}^{\text{intended}})$ using the following two dimensions:

**Relationship Between Signatures:**
1. $\sigma(T) \subset \sigma(\text{Th}(\mathcal{M}^{\text{intended}}))$
2. $\sigma(\text{Th}(\mathcal{M}^{\text{intended}})) \subset \sigma(T)$
3. $\sigma(T) \iff \sigma(\text{Th}(\mathcal{M}^{\text{intended}}))$
4. $\sigma(T) \parallel \sigma(\text{Th}(\mathcal{M}^{\text{intended}}))$

**Relationship Between Axioms:**
1. $T \models \text{Th}(\mathcal{M}^{\text{intended}})$, $\text{Th}(\mathcal{M}^{\text{intended}}) \not\models T$
2. $\text{Th}(\mathcal{M}^{\text{intended}}) \models T$, $T \not\models \text{Th}(\mathcal{M}^{\text{intended}})$
3. $T \iff \text{Th}(\mathcal{M}^{\text{intended}})$
4. $T \parallel \text{Th}(\mathcal{M}^{\text{intended}})$

For each scenario, we identify the necessary reuse operation(s) required and prove their correctness. The types of reuse required for each case are summarized in Table 5.1.

Certain scenarios are logically impossible, for example Case 1 describes an instance where the candidate theory $T$ has signature that is a subset of the signature of $\text{Th}(\mathcal{M}^{\text{intended}})$ - it is clearly impossible for $T$ to entail $\text{Th}(\mathcal{M}^{\text{intended}})$ in this situation as there must be terms and therefore sentences in $\text{Th}(\mathcal{M}^{\text{intended}})$ that are not expressible in $T$. Cases 1, 3, 6, 7, 13, 14, and 15 are not possible in similar ways; thus we need consider only those remaining to prove Theorem 7.

Of the remaining possible cases, we consider them based on the strength of the candidate ontology, relative to the required ontology. By Theorem 5, the order of operations is irrelevant, however to avoid temporary inconsistencies, we adopt a convention of applying Extraction operations first.

**Candidate is ‘Just Right’**
By definition, \( T \) can be reused via the As-Is operation for \( Th(M_{\text{intended}}) \).

\[
\text{as.is}(T) = T = Th(M_{\text{intended}})
\]

### Candidate Weaker than Required

**Case 2:**  \( \sigma(T) \subset \sigma(Th(M_{\text{intended}})) \) and \( Th(M_{\text{intended}}) \models T, T \not\models Th(M_{\text{intended}}) \)

By definition, the Extension operation must be applied. Precisely which Extension subtype depends on the modules of \( Th(M_{\text{intended}}) \).

- If there exists a subset of the required axiomatization \( Th(M_{\text{intended}})^t \subset Th(M_{\text{intended}}) \)
  such that \( \sigma(Th(M_{\text{intended}})^t \cap Th(M_{\text{intended}})^t') \not\subset \sigma(T) \)
  and \( \sigma(Th(M_{\text{intended}})^t') = \sigma(T) \) and \( T = Th(M_{\text{intended}})^t' \),
  then \( T \) satisfies a module of a particular domain in \( Th(M_{\text{intended}}) \) and by definition we can apply a Domain Extension to achieve \( Th(M_{\text{intended}}) \) from \( T \).
  There exists a \( T^+ \) such that \( \text{domain.extension}(T, T^+) = Th(M_{\text{intended}}) \).

- Else, if there exists a subset of the required axiomatization \( Th(M_{\text{intended}})^t \subset Th(M_{\text{intended}}) \)
  such that \( \sigma(Th(M_{\text{intended}}) \cap Th(M_{\text{intended}})^t') \not\subset \sigma(T) \)
  and \( \sigma(Th(M_{\text{intended}})^t') = \sigma(T) \) and \( Th(M_{\text{intended}})^t' \models T \) (and \( T \not\models Th(M_{\text{intended}}) \)),
  then \( T \) partially satisfies a module of a particular domain in \( Th(M_{\text{intended}}) \).
  By definition we must apply a Strengthening Extension and a Domain Extension to achieve \( Th(M_{\text{intended}}) \) from \( T \).
  There exists some \( T^+, T_{sub}^+ \) such that \( \text{domain.extension(strengthening_extension}(T, T^+), T_{sub}^+) = Th(M_{\text{intended}}) \).

- Otherwise, by definition a Strengthening Domain Extension is required (i.e., strengthening within and beyond the domain of \( T \)).
  There exists some \( T^+ \) such that \( \text{strengthening.domain.extension}(T, T^+) = Th(M_{\text{intended}}) \).
An example of this case is in a scenario where we’re reusing an ontology that covers only part of the scope we require. For example, say we are designing an event ontology and wish to reuse some existing work. We might be able to reuse an existing event ontology that partially describes the behaviour of events, despite being weaker than we require and perhaps omitting some concepts of interest such as the notion of a sub-event.

### Case 10: \( \sigma(T) \iff \sigma(Th(M^{intended})) \) and \( T \vdash Th(M^{intended}) \)

By definition, the **Strengthening Extension** operation is required to reuse \( T \) for \( Th(M^{intended}) \).

There exists some \( T^+ \) such that \( \text{strengthening_extension}(T, T^+) = Th(M^{intended}) \)

An example of this case is in a scenario where the ontology to be reused has a larger scope than we require. For example, if we are reusing an upper ontology such as DOLCE \([22]\) to develop a theory of Participation, we only require some of the ontology because its scope includes many other, unrelated concepts such as the notion of a Physical Feature or an Amount of Matter.

### Candidate Stronger than Required

#### Case 5: \( \sigma(Th(M^{intended})) \subset \sigma(T) \) and \( T \vdash Th(M^{intended}), Th(M^{intended}) \not\vdash T \)

By definition, the **Extraction** operation must be applied. Precisely which Extraction subtype depends on the modules of \( T \).

- If there exists a subtheory \( T' \subset T \) such that \( \sigma(Th(M^{intended})) = \sigma(T') \)
  \( \sigma(T' \cap T) \not\subset \sigma(Th(M^{intended})) \)
  and \( T' = Th(M^{intended}) \)
  then a module of \( T \) satisfies \( Th(M^{intended}) \). By definition, **Domain Extraction** is must be applied to reuse \( T \) for \( Th(M^{intended}) \).
  There exists a \( T^- \) such that \( \text{domain_extraction}(T, T^-) = Th(M^{intended}) \).

- Else, if there exists a subtheory \( T' \subset T \) such that \( \sigma(Th(M^{intended})) = \sigma(T') \),
  \( \sigma(T' \cap T) \not\subset \sigma(Th(M^{intended})) \), and
  \( T' \vdash Th(M^{intended}) \) (and \( Th(M^{intended}) \not\vdash T' \)) then a module of \( T \) satisfies, but is stronger than the requirements. By definition, **Domain Extraction and Weakening Extraction** is must be applied to reuse \( T \) for \( Th(M^{intended}) \).
  There exists a \( T^-, T^-_{sub} \) such that \( \text{weakening_extraction}((\text{domain_extraction}(T, T^-)), T^-_{sub}) = Th(M^{intended}) \).

- Else, by definition a **Weakening Domain Extraction** is required to reuse \( T \) for \( Th(M^{intended}) \).
  There exists a \( T^- \) such that \( \text{weakening_domain_extraction}(T, T^-) = Th(M^{intended}) \).
Case 9: \( \sigma(T) \iff \sigma(Th(M^{intended})) \) and \( T \models Th(M^{intended}) \). There exists a \( T^- \) such that \( \text{weakening}\_\text{extraction}(T, T^-) = Th(M^{intended}) \).

By definition, the \textit{Weakening Extraction} operation is required to reuse \( T \) for \( Th(M^{intended}) \).

Candidate Incomparable to Required in other words, \( T \) is stronger than required in some ways, but weaker than required in others. By definition, these cases will require an \textit{Extraction} operation and an \textit{Extension} operation.

Case 4: \( \sigma(T) \subset \sigma(Th(M^{intended})) \) and \( T \parallel Th(M^{intended}) \).

The precise Extraction operation required depends on the nature of the conflict between \( T \) and \( Th(M^{intended}) \).

- If there exists a subtheory \( T' \subset T \) such that \( \sigma(T) = \sigma(T') \) and \( Th(M^{intended}) \models T' \) then by definition, a \textit{Weakening Extraction} is required.
  
  There exists a \( T^- \) such that \( \text{weakening}\_\text{extraction}(T, T^-) = T' \subset Th(M^{intended}) \).

- Else, if there exists a subtheory \( T' \subset T \) such that \( \sigma(T/T') \not\subset \sigma(T') \) (i.e. \( T' \) is a module of \( T \), \( T \) conservatively extends \( T' \) and \( Th(M^{intended}) \models T' \) then an entire module of \( T \) is in conflict. By definition, \textit{Domain Extraction} is required.
  
  There exists a \( T^- \) such that \( \text{domain}\_\text{extraction}(T, T^-) = T' \subset Th(M^{intended}) \).

- Else, by definition \textit{Weakening Domain Extraction} is required.
  
  There exists a \( T^- \) such that \( \text{weakening}\_\text{domain}\_\text{extraction}(T, T^-) = T' \subset Th(M^{intended}) \).

The application of any of the above operations results in the reduction of Case 4 to Case 2: \( \sigma(T') \subset \sigma(Th(M^{intended})) \) and \( Th(M^{intended}) \models T' \). Thus, an \textit{Extension} operator is required subsequently, as identified in Case 2, to reuse \( T' \) for \( Th(M^{intended}) \).

An example of this case is a scenario where we are reusing a theory that partly covers the required scope and captures the semantics we require in some ways, but not in others. Consider again the development of an ontology of events. Say we are to reuse a theory of dense time, but our ontological commitment is one of discrete time which is in fact inconsistent with the semantics of dense time. However, the ontology of dense time still possesses semantics that are useful for our ontology (specifically, those of a linear ordering). We can reuse it by weakening its semantics so that it is no longer incomparable, and then extending it to capture the required ontological commitments and scope of concepts.

Case 8: \( \sigma(Th(M^{intended})) \subset \sigma(T) \) and \( T \parallel Th(M^{intended}) \).

Some of the signature of \( T \) must be removed, therefore there are two possible extraction operations that will be required, depending on the nature of the modules of \( T \).

- If there exists a subtheory \( T' \subset T \) such that \( \sigma(T/T') \not\subset \sigma(T') \) (i.e. \( T' \) is a module of \( T \), \( T \) conservatively extends \( T' \) and \( Th(M^{intended}) \models T \) then by definition, \textit{Domain Extraction} is required.
  
  There exists a \( T^- \) such that \( \text{domain}\_\text{extraction}(T, T^-) = T' \subset Th(M^{intended}) \).
• Else, Weakening Domain Extraction is required.

There exists a $T^-$ such that $\text{weakening\_domain\_extraction}(T, T^-) = T' \subset T h(M^{\text{intended}})\) 

The precise Extension operation required depends on the nature of the conflict between $T'$ and $T h(M^{\text{intended}})$. The conflict dictates the effect of the required extraction, not only on the semantics of the resulting $T'$, but its signature $\sigma(T')$.

• If $\sigma(T') = \sigma(T h(M^{\text{intended}}))$ then the extraction has resulted in the reduction of Case 8 to Case 10: $\sigma(T') \iff \sigma(T h(M^{\text{intended}}))$ and $T h(M^{\text{intended}}) \models T', T' \not\models T h(M^{\text{intended}})$. By definition, the Strengthening Extension operation is required to reuse $T'$ for $T h(M^{\text{intended}})$.

There exists some $T^+$ such that $\text{strengthening\_extension}(T', T^+) = T h(M^{\text{intended}})\).

• Else $(\sigma(T') \subset \sigma(T h(M^{\text{intended}}))$ then the extraction has resulted in the reduction of Case 8 to Case 2: $\sigma(T') \subset \sigma(T h(M^{\text{intended}}))$ and $T h(M^{\text{intended}}) \models T'$. Thus, an Extension operator is required subsequently, as identified in Case 2, to reuse $T'$ for $T h(M^{\text{intended}})\).

**Case 12:** $\sigma(T) \iff \sigma(T h(M^{\text{intended}}))$ and $T \parallel T h(M^{\text{intended}})$

This case is the most complex and it something of a combination of Case 4 and Case 8. The precise Extraction operation required depends on the nature of the conflict between $T$ and $T h(M^{\text{intended}})$. If there exists a subtheory $T' \subset T$ such that $T h(M^{\text{intended}}) \models T'$ then by definition, Weakening Extraction is required.

There exists a $T^-$ such that $\text{weakening\_extraction}(T, T^-) = T' \subset T h(M^{\text{intended}})\).

• Else, if there exists a subtheory $T' \subset T$ such that $\sigma(T/T') \not\subseteq \sigma(T')$ (i.e. $T'$ is a module of $T$), $T$ conservatively extends $T'$ and $T h(M^{\text{intended}}) \models T'$ then by definition, Domain Extraction is required.

There exists a $T^-$ such that $\text{domain\_extraction}(T, T^-) = T' \subset T h(M^{\text{intended}})\).

• Else, by definition Weakening Domain Extraction is required.

There exists a $T^-$ such that $\text{weakening\_domain\_extraction}(T, T^-) = T' \subset T h(M^{\text{intended}})\).

The precise Extension operation required also depends on the nature of the conflict between $T$ and $T h(M^{\text{intended}})$ (i.e. the extraction that was necessary).

• If $\sigma(T') = \sigma(T h(M^{\text{intended}}))$ then the extraction has resulted in the reduction of Case 8 to Case 10: $\sigma(T) \iff \sigma(T h(M^{\text{intended}}))$ and $T h(M^{\text{intended}}) \models T, T \not\models T h(M^{\text{intended}})$. By definition, the Strengthening Extension operation is required to reuse $T$ for $T h(M^{\text{intended}})$.

There exists a $T^+$ such that $\text{strengthening\_extension}(T', T^+) = T h(M^{\text{intended}})\).

• Else $(\sigma(T') \subset \sigma(T h(M^{\text{intended}}))$ then the extraction has resulted in the reduction of Case 8 to Case 2: $\sigma(T) \subset \sigma(T h(M^{\text{intended}}))$ and $T h(M^{\text{intended}}) \models T$. Thus, an Extension operator is required subsequently, as identified in Case 2, to reuse $T$ for $T h(M^{\text{intended}})\).

**Case 16:** $\sigma(T) \parallel \sigma(T h(M^{\text{intended}}))$ and $T \parallel T h(M^{\text{intended}})\)

This case is similar to Case 12 with the exception that due to the signature overlap, modifications to the domain are required for both Extraction and Extension. The precise Extraction operation required depends on the nature of the conflict between $T$ and $T h(M^{\text{intended}})$.  


• If there exists a subtheory \( T' \subset T \) such that \( \sigma(T/T') \not\subseteq \sigma(T') \) (i.e. \( T' \) is a module of \( T, T \) conservatively extends \( T' \) and \( Th(\mathfrak{M}_{\text{intended}}) \models T' \) then by definition, Domain Extraction is required.  
There exists a \( T' \) such that domain\_extraction\( (T, T') = T' \subset Th(\mathfrak{M}_{\text{intended}}) \)

• Else, by definition Weakening Domain Extraction is required.  
There exists a \( T' \) such that weakening\_domain\_extraction\( (T, T') = T' \subset Th(\mathfrak{M}_{\text{intended}}) \)

The application of the Extraction operation results in the reduction of Case 16 to Case 2: \( \sigma(T) \subset \sigma(Th(\mathfrak{M}_{\text{intended}})) \) and \( Th(\mathfrak{M}_{\text{intended}}) \models T. \) Thus, an Extension operator is required subsequently, as identified in Case 2, to reuse \( T \) for \( Th(\mathfrak{M}_{\text{intended}}). \) The precise Extension operation required also depends on the nature of the conflict between \( T \) and \( Th(\mathfrak{M}_{\text{intended}}). \)

**Reuse of Multiple Theories:** Now we consider the reuse of multiple ontologies \( T_1, ..., T_n \) for \( Th(\mathfrak{M}_{\text{intended}}). \) As with the reuse of a single theory, while the order of reuse operations does not affect the outcome, we consider the application of the Extraction operation(s) first. This must be done initially, before the Combination of the theories to avoid potential inconsistencies that would prohibit the identification of reuse cases. This allows the identification of operations to be made more precise, with respect to the individual theories.

**Extraction** For each \( T_i \), the individual reuse cases may be identified as in the previous section, with respect to \( Th(\mathfrak{M}_{\text{intended}}) \), as shown for the reuse of a single ontology. Any required instances of Extraction that are identified will be required for the reuse of \( T_1, ..., T_n \) for \( Th(\mathfrak{M}_{\text{intended}}). \)

The assessment of Extension operations at this stage cannot be identified accurately with respect to the individual theories because this would not account for the expansion of signature and/or strengthening of semantics that will result from the Combination operations.

**Combination** Following any applicable Extraction operations, the theories \( T'_1, ..., T'_n \) will then be combined. Precisely which Combination operators are required will vary depending not only on \( \mathfrak{M}_{\text{intended}} \) but on the context of the operation, however it is a given that in this case all of the theories will be combined.

For example, if we are combining two theories \( T_1, T_2 \) and \( \sigma(T_1) \subset \sigma(T_2) \) then for \( T_2 \) it may be a Strengthening Combination with \( T_1 \), whereas for \( T_1 \) it might be a Domain Strengthening Combination with \( T_2 \).

The combination of the theories results in a single theory that can then be assessed to determine the final required operations.

**Extension** For the combined theory, \( T = T'_1 \cup ... \cup T'_n \), it is straightforward to assess the required operations as it reduces to the cases identified for the reuse of a single theory. Following the Extraction and Combination operations, the required Extension operation(s) can now be diagnosed. Since the necessary extraction operations will have already been applied, it is straightforward to see that we must have either \( \sigma(T) \subset \sigma(Th(\mathfrak{M}_{\text{intended}})) \) or \( \sigma(T) = \sigma(Th(\mathfrak{M}_{\text{intended}})) \), and \( Th(\mathfrak{M}_{\text{intended}}) \models T, T \not\models Th(\mathfrak{M}_{\text{intended}}) \) or \( T \iff Th(\mathfrak{M}_{\text{intended}}). \) Therefore, \( T \) will be identified as one of Case 2, Case 10, or Case 11 and the required operations may be diagnosed as previously outlined.

The specialized operations are defined such that they indicate the implications for shareability with the reused theories; if an operation results in a non-conservative extension of the original (input) theory, or visa versa,
we can infer that shareability will not be possible between the two. The proof of Theorem 4 has served not only to demonstrate the completeness of the specialized operations as a means of characterizing reuse, it also demonstrates a means by which these operations can be identified in any possible scenario of reuse. This analysis may be employed by ontology developers in order to identify the necessary specialized operations, which will in turn inform developers of the implications that reuse of a particular theory will have on its semantics, and consequently the resulting shareability that can be expected. Note that as with the earlier results on the reduction of design work (Section 5.3), we observe that to implement these results in practice would require the use of an approximation of $\text{Th}(\mathfrak{M}_{\text{intended}})$.

In the case that multiple reuse operations are necessary, the assessment of the resulting shareability follows easily: any operation that does not preserve the semantics of $T$ would imply that overall, its semantics are not preserved. While the case of multiple ontologies requires special consideration for the operations (and order) to be applied, logically the relationship between a given theory and the final, required ontology will be the same. Therefore, even in cases of combinations we only need to assess each theory of the set individually to determine whether the benefits of shareability can be expected with its reuse. This is because the question of preservation of semantics is relative to each theory, not the entire set of theories. The result serves to inform the developer of the shareability between the resulting ontology and the reused theories. It also contributes even more precise guidance regarding the operations required, thus further enabling the reducing of design work.

5.4.3 On the Uniqueness of Necessary Operations

The property of uniqueness would be particularly valuable for the developer – not only in directing the performance of reuse, but in interpreting the shareability implications for any given reuse scenario. Uniqueness allows for the removal of any uncertainty with respect to the necessary operations: is there any other way to go about this? Must I lose shareability with this entire theory? For example, say we are considering the reuse of two theories, in combination. If we identify that reuse via a strengthening domain combination is necessary, with the knowledge that this result is unique then we need only assess whether the loss of shareability is acceptable. If this result is not necessarily unique, then the developer should also consider whether there are any alternative approaches that might offer improved results. It is useful to know, especially when considering multiple possible reuse scenarios, that the result that has been found is the only one possible. Further, knowledge of a unique solution would prove valuable in guiding the reuse itself during the Design of the chosen ontologies.

We speculate that the approaches described here identify the minimal set of necessary operations, and that these operations are in fact unique.

**Conjecture 1.** *There exists a unique set of necessary reuse operations (and specialized operations) for the reuse of any $T_1, ..., T_n$ to axiomatize any $\mathfrak{M}_{\text{intended}}$.*

Given that each identified reuse case has a distinct outcome (specialized or otherwise), it would appear that the necessary set produced must in fact be minimal. In practice, we may reuse some ontology $T$ via $\text{extension}(T, T')$ and then extend the result of this: $\text{extension}(\text{extension}(T, T'), T'')$ whereas we could have obtained the same result with a single application: $\text{extension}(T, T' \cup T'')$. Certainly, there are many possible such variations that achieve the same resulting axiomatization, and by definition all of these alternatives may be found to be necessary with respect to some set of operations. However, consider the fact that in the proofs presented here, the reuse of any theory(s) is characterized by at most 2 operations, it would seem that there must exist some minimal set of operations that achieves reuse in the most efficient way. Further, it would seem that this minimal set is indeed
identified by way of the approach described here. Then, assuming that each reuse operation is disjoint from the
others, it would seem that there should be only one such minimal set.

It is fairly trivial to see that the reuse operations are distinct. Adding a theory not in a repository (extension)
is certainly not the same as removing a theory, nor could we hope to define one by the other. Similarly, we
cannot hope to characterize the combination of a theory that is in a repository via the addition of axioms that
are not in a repository. While this distinction is less clear for the specialized reuse operations, closer inspection
indicates that these too are disjoint. For example, consider the specialized extension operations. A strengthen-
ing_extension, by definition cannot be a domain_extension because the additional axioms ($T^+$) added for each
specialized operation necessarily have a disjoint signature: in one case the signature may be equivalent or a subset
of the theory $T$ being reused, whereas in the other case is must be disjoint from this signature. With similar ratio-
nale, no strengthening_extension may be expressed as a strengthening_domain_extension as the first specialized
operation requires the addition of axioms with a signature distinct from the second: $\sigma(T^+) \subseteq \sigma(T)$ compared to
$\sigma(T^+) \supseteq \sigma(T)$. The only suspect may be the potential relationship between the strengthening_domain_extension
and the strengthening_extension with the domain_extension. However, we find that the definition of the former is
specified such that it cannot be even partially expressed as either of the latter. The conditions on the axioms added
in a strengthening_domain_extension preclude even a subtheory to be characterized as a strengthening_extension
or a domain_extension.

5.5 Discussion

Beyond their obvious theoretical importance, in this chapter we employed the definitions of reusability and reuse
as tools to achieve the objective of tractable benefits of reuse. In this section, we consider additional ways in which
the definitions presented here can be leveraged. The definition of reuse specified in this Chapter also provides the
means to recognize and describe issues with reuse in the current state. Further, it provides a language to describe
how reuse of ontologies is related to other concepts, leaving us better equipped to study and make progress in the
area of reuse.

5.5.1 Current Issues with Reuse

In addition to supporting the task of reuse, the provision of a formal definition facilitates the identification of issues
in current practice. In general, given that we have identified several distinct reuse operations, each with varying
logical implications, it should be clear that to reference ‘reuse’ of another theory alone is inadequate in terms of
design documentation. Without knowing whether the reused concepts of some ontology, $T_1$ are conservatively,
non-conservatively, (or perhaps neither) extended by the new ontology, $T_2$, how are we to interpret this notion of
‘reuse’? How should we know whether this $T_2$ is capable of semantic integration with other ontologies that have
reused $T_1$?

Apart from inhibiting the ease with which the benefit of shareability can result from reuse, the current im-
precise use of the term reuse introduces serious risks of miscommunication, and encourages confusion regarding
the task of reuse and its outcomes. Specifically, we identify two such issues related to a misuse of International
Resource Identifiers (IRIs).
Inaccurate Use of IRIs

In the standard languages for ontology encoding, all of the terms in an ontology have a unique IRI. In OWL and RDF, these IRIs are typically abbreviated with namespaces, whereas in Common Logic they are inherited via the document’s IRI. These identifiers are a key mechanism for interoperability and reuse, as they provide a means of referencing and disambiguation when including terms from one or more existing ontologies. They facilitate interoperability, as IRI-matching can identify when two ontologies are referring to the same concept. Further, from a documentation standpoint, these identifiers provide developers with a means of referencing the ontologies that have been reused, and indicating where this reuse has occurred. The formal definition of reuse and classification of reuse operations presented here highlights a serious issue with the current use of these IRIs in ontology development; current practice renders these identifiers both misleading and incapable of providing the aforementioned support for ontologies.

Currently, the original IRI is maintained when a concept is reused. In the context of linked data [6], it makes sense to use the IRI to indicate the concept being referred to. On the other hand, we’ve seen that the reuse of an ontology can occur in several different ways, each with different logical implications; for example, concept(s) may be non-conservatively extended (or even made to be inconsistent). This means that a concept, with the same IRI, may take on a new meaning when it is reused in the development of another ontology. This possibility corrupts the integrity of the IRI. While it may be a reliable indicator of provenance, it is no longer an accurate means of disambiguation or tool for integration. For example, consider the following scenario: in two separate occasions we perform strengthening extensions on ontology $T_1$ for the creation of ontologies $T_2$ and $T_3$. According to current practice, in both $T_2$ and $T_3$ we would have some terms with the same IRI, yet different (stronger) semantics than in the original, $T_1$. Now, consider an even more confusing scenario: the extensions made to the semantics in $T_2$ and $T_3$ are inconsistent in some way. We now have a single term, named with a single IRI, with two inconsistent meanings. As an outsider intending to use or reuse any of these ontologies, how can we interpret this? Worse still, consider the possible consequences should a potential user work under the assumption that each of these ontologies shares a single meaning of this identically named concept.

Given that the role of an IRI is to provide a unique identification for a specific concept, we should expect the semantics of the concept to be the same in all of its instances. A concept with different semantics - whether stronger, weaker, or unrelated - is a different concept, thus we should expect it to have a different IRI. It follows from this that any reuse of an existing module of an ontology must be a conservative extension if the IRIs of the module are to be maintained; in other words, if the semantics of a concept changes, then its IRI should also be revised.

One final consideration in this solution is in maintaining the use of the IRI for provenance documentation regarding the ontologies reused. It is important not to lose information regarding the source, while still making the distinction that the semantics have changed. Various solutions are possible for this, while still resolving the issue described previously. For example, in cases where the reuse strengthens a particular concept, the original term and IRI might be maintained, and a new term could be added (e.g. as a subclass) to specify the stronger semantics. Regardless of the solution to the question of provenance, it is clear that the current use of IRIs is an impediment to the vision of reuse and must be resolved.

The reuse operations identified in this chapter provide a tool to identify this issue when it arises. In any instance of reuse where the combination of operations do not preserve the semantics of the original theory, it is clear that the same IRI cannot be used in the resulting ontology. In this case, as we have recognized above, there are essentially two options: (1) the developer can create a new IRI to distinguish the affected terms, or (2) if the required ontology is a non-conservative extension of the original ontology, the developer can adjust their
requirements to allow the required ontology to contain both the original and modified concepts in such a way that the required operations will preserve the semantics of the original theory. Note that this second option is not possible in the case that the original ontology is a non-conservative extension of the required ontology, nor is it possible if the two theories are incomparable.

**Term Reuse**

One reason that the former issue has gone largely unnoticed until now can be found in a generally accepted, but faulty practice of reusing individual terms. More precisely, in current practice we find terms from existing ontologies are “reused” by referencing the IRI of the term alone, without the import of part or all of its originating ontology. Examples of this practice are commonplace; it can be observed in many well-established and well-known ontologies, and it has even been described as a preferred practice of ontology developers (i.e. as opposed to use of the imports function) in [73].

We find this sort of IRI-based reuse is a technique prescribed by the linked data principles [6]; it is essential to the paradigm of linked data, and we are by no means criticising it in that context. The error occurs in extending this practice directly for ontology design. Reusing terms in this simplistic way provides an inaccurate representation of their semantics. These IRI references are interpreted in the same way (as defined in XML) throughout the languages in the Semantic Web. They provide a unique identifier for the concept which, according to linked data principles should be dereferencable to information describing the concept. If a single term is reused in the development of a new ontology, this information is simply available, it is not deduced from the IRI itself. Certainly, it is possible to dereference the IRI and review or import the axioms that contribute to the concept's definition, however the inclusion of a single IRI only indicates where these axioms may be found, it does not imply their inclusion wherever the term is used. Thus, including a term in this way is nothing more than a reference to some external concept, with no bearing on the semantics of the ontology being designed. A clear distinction must be made between the reuse of terms, the specification of equalities/mappings that add semantics to the ontology, and the specification of mappings that can be inferred from the existing axioms alone. In the former two cases, the associated semantics must be imported: either we must modify OWL2 and its tools such that an external IRI reference is to be interpreted as importing the source ontology; or, it should be a requirement that in order to reuse a term via IRI reference, the source ontology must already be imported.

Clearly, issues may arise in some cases where the source ontology is quite large and many of the axioms are not required in the new ontology. Ideally, we would like to import only the axioms that are relevant to the concept being reused. However, this is an issue that must be addressed by modularity, not through meaningless references disguised as reuse.

### 5.5.2 Classification and DOL

The Distributed Ontology Language (DOL) [46], has been designed in response to the OntoIOP request for proposal [59] to address the challenge of interoperability for heterogeneous formal representations. While the scope of this project is much broader and not quite aligned with the work presented here, we have found that the reuse operations identified here correspond in some ways to the ‘structuring language for OMS’ provided by DOL. Should DOL be approved as a standard, this has the potential to be beneficial for our solutions as it increases the likelihood of infrastructure support for the documentation of the different types of reuse as well as the implementation of the associated solutions presented in this chapter.

The DOL project considers not only ontologies, but ‘specifications and MDE [Model-Driven Engineering]
models’ in a variety of languages, and is also focused not on reuse but on integration and interoperability. The scope of the definition of reuse presented here applies to a more restricted context, therefore we find that while we are able to identify some relationships between each reuse operation, and an OMS term in DOL, we cannot conversely identify relationships between each OMS term and a type of reuse. The following relationships have been identified based on the natural language definitions of the DOL terms defined in Section 4.3 ‘Structured OMS’ of the standard submission dated: Monday 24th August, 2015.

**domain_extraction** relates to the OMS construct approximation with the use of the forget operation, in order to remove a subtheory that is nonconservatively extended by the original ontology. Intuitively, extraction construct may seem more appropriate as it extracts a module of the ontology, however in DOL a module is defined as a subset as opposed to a subtheory of the axioms.

**weakening_extraction** requires the result to have the same signature as the original theory; this condition is not something that is captured by the OMS terms, however this sort of reuse is more generally captured (and could be expressed) by the filtering construct.

**weakening-domain_extraction** again, this operation is captured by the OMS construct filtering, with the exception of the condition on the resulting signature.

**domain_extension** this operation is captured by the extension OMS construct, specified to be a Conservative extension.

**strengthening_extension** this operation is captured partially by the OMS construct extension. While the structured OMS specifies many variations on extensions, none captures the more specific notion of a strengthening_extension – in particular, the condition that the resulting signature be unchanged, while the semantics are strengthened.

**strengthening_domain_extension** this operation corresponds to the OMS construct extension. As with the strengthening_extension, although a strengthening_domain_extension may be expressed using the extension construct, it is unable to capture the constraint on the resulting signature.

**domain_combination** this operation corresponds to the union OMS construct, specified to be a Conservative extension.

**strengthening_combination** while intuitively, this operation corresponds to the OMS construct union, according to DOL members of unions must be self-contained, therefore we can only express a strengthening_combination as a kind of extension OMS. As with the strengthening_extension, the distinction regarding the resulting signature is not captured by this OMS construct. The criteria for a union in DOL are clearly distinct from those of the combination reuse operation.

**strengthening_domain_combination** similarly, although the distinction regarding the resulting signature is not captured, we can express a strengthening_domain_combination as a kind of extension OMS construct.

Each of the reuse operations defined here may be expressed by some DOL construct. However certain distinctions such as the difference between the original and resulting signature that are made when considering operations in the context of reuse, are not captured by these constructs as they are simply not relevant when defining metalogical relationships between theories. As mentioned, the DOL constructs provide a means of describing the structure of an ontology such that we can understand its relationship to other ontologies. A key distinction is that the
language in DOL defines constructs, which are meant to describe the structure of theories and how they relate to one another, as opposed to the definition of operations here which represents a manipulation performed on a given ontology. The concept of an ‘original’ (reused) ontology, or even the source of axioms (another ontology, or created by the developer) make little sense from this perspective.

5.6 Summary

In this chapter, we presented a formal, detailed definition of reuse and reusability. In order for the task of ontology development to move from an art to an engineering discipline, technical concepts such as ontology reuse must be well-defined. This is a contribution in its own right, as no such definition exists in the literature to-date. Better still, we leverage these definitions to address the second objective of this thesis. We clearly demonstrate how both of the supposed benefits of reuse can be achieved, thereby providing ontology developers with greater control over the attainment of these benefits.

Owing to the detail of our definitions, we are able to provide guidance and feedback for the developer that may be derived in any case of reuse. The developer can identify how - specifically with what operations - a theory or set of theories can be reused to create the required theory. This further reduces the work required for reuse. Building on these results, the developer can also assess whether or not a theory’s semantics will be preserved, and thus be well-informed about whether it will attain the benefit of interoperability (and for which theories) prior to performing reuse.

While these results alone are sufficient to facilitate the attainment of the benefits of reuse, the proof of Theorem 7 from which both the design guidance and interoperability assessment are derived clearly has a decision tree structure. It would be straightforward to implement these results in a procedure, thus further simplifying the analysis for the developers, and making the assessment of multiple alternatives much more pragmatic. This would be a very useful direction for future work. We also proved a property regarding the ordering of reuse operations. Assuming that an automated procedure is pursued, future work considering other properties of reuse, such as transitivity, may also be useful.

Finally, an important observation to be made is on the importance of modularity for the results in this chapter. The identification of scenarios, and assessment of reuse outcomes relies heavily on the identification of modules within the theories. Of course, this is something that may be computed after-the-fact, however the practicality of this approach relies on the modularization of the theories thus we add this to the chorus of motivating scenarios for the design principle of ontology modularity.
Chapter 6

Examining the Benefits of Reuse in Practice

In the previous chapter, we presented a definition of reusability and reuse. We then described how these results could be used to make the benefits of reuse more tractable through the recognition of reuse cases and the necessary operations for the reuse of a given candidate. As we saw, it's important to be able to identify the different cases as different reuse operations affect the degree to which the benefits of reuse are achieved. In this chapter we demonstrate this with the analysis of two ontologies that were developed from different cases of ontology reuse. The Date Ontology, presented in [40] reuses several existing theories and was developed with the aim of completely capturing the intuitions of dates, without making any unnecessary assumptions (e.g. committing to a particular calendar). The Home Services Ontology (HSO) [48] was developed with the aim of supporting front-end functionality for a website offering home improvement services.

These ontologies illustrate the considerable impact of reuse operations on the benefits of both the design work required for reuse, as well as shareability of the resulting ontology. While both ontologies are good examples of reuse in practice, we will see that owing to the reuse operations that were performed, the Dates ontology demonstrates all of the benefits of reuse, whereas the HSO does not. The value of the solution presented in Chapter 5 is that it provides the necessary foundation in order to make useful predictions and assessments regarding such outcomes for design via reuse given a particular candidate(s).

6.1 An Ontology of Dates

Reasoning about dates is a fundamental capability required for many applications, from supply-chain management and e-commerce to narrative analysis. For ontologies to support these sorts of applications they need to be able to infer, for example, that an order is late if we know that today is Friday and the order was sent on Monday, but the order should take only two days to arrive.

Earlier approaches to ontologies for durations, dates, and calendars lack complete axiomatizations – the intended semantics is specified in documentation but there are insufficient axioms to guarantee that all models are intended, or else arithmetic is used as the intended model (that is, reasoning about time durations reduces to reasoning about real numbers). The Dates ontology presented in [40] was developed to address this with a first-order

\[\text{Note that an extended version of this original publication containing proofs of the representation theorems stated here is available at:} \]

theory that makes a minimal set of assumptions to axiomatize the class of intended models.

6.1.1 Dates in Practice

Although it is tempting and even intuitive to consider a date to simply be a way of naming and referring to particular time points, it is easy to see that they are much more than that. One way to understand this is to look at how we think about and use the concept in practice. Dates contain much more information than simply their relative order on a timeline. Consider all of the knowledge that is implicitly captured in the date: Monday, October 20. We are confronted with an intuition of repeatability – we know that there was a Monday before this Monday, and that there will be another Monday a week after it. We know that not only did this particular Monday take place in October, but so did the previous (and so will the following) Monday. Further, we know that the October we are referring to in this date was part of a year. Our ability to make such inferences is based on two key properties of dates: periodicity and composition. In order to represent and reason about dates correctly, taking full advantage of this sort of knowledge, these properties need to be captured.

An additional characteristic stems from an important distinction which must be made between the date concepts that are the focus of this ontology, such as “Monday”, “week”, “year” and the dates that we encounter in day-to-day conversation, such as “Monday, October 20”. Whereas the properties of periodicity and composition are more or less observable in both, there is a stronger semantics that might easily be missed if our understanding of the distinction between these concepts is not clear. There is, in fact, a meaningful relationship between the concepts of date and time that must be captured to accurately represent the semantics of dates. Consider for example, a particular day like today. Today has some date, which we know will correspond to a particular day of the week. Further, we know that this date must correspond to a unique day of the week. We also know that dates have specific start and end times – today, whichever day it is, started at midnight and will end at 11:59 P.M. We can construct similar examples to illustrate this semantics for other calendar constructs (Gregorian or otherwise), such as months.

These observations can be condensed into three key characteristics of dates: composition, periodicity, and repeatable occurrences (with unique start and end times). The Date Ontology captures all of these characteristics in a simple and effective manner.

6.1.2 Using the Date Ontology

The Date Ontology defines the concepts of dates independent of any given calendar. In this way, the Date Ontology can be used to represent and reason about any particular calendar by adding a domain theory to define the calendar’s constructs. In the following example, we illustrate a portion of the axiomatization of such a domain theory for the Gregorian calendar.

The composition of days into weeks – Monday is part of a week, as is Tuesday:

\[ \text{subactivity}(\text{Monday}, \text{Gregorian\_week}) \]
\[ \text{subactivity}(\text{Tuesday}, \text{Gregorian\_week}) \]

The duration aspect of periodicity – Mondays and Tuesdays are always a day long, and they occur at weekly intervals:

\[ (\forall o) \text{ occurrence\_of}(o, \text{Monday}) \supset \text{duration}((\text{beginof}(o), \text{endof}(o)) = \text{Day} \]
\[ (\forall o) \text{ occurrence\_of}(o, \text{Tuesday}) \supset \]
duration((beginof(o), endof(o))) = Day
freq(Monday, Week)
freq(Tuesday, Week)

The ordering aspect of periodicity – Monday comes before Tuesday in every week:

\((\forall o) \text{ occurrence of } o, \text{Gregorian week} \supset\)
\((3s_1, s_2) \text{ leaf}(s_1, Monday) \land \text{ root}(s_2, Tuesday)\)
\land \text{ min_precedes}(s_1, s_2, \text{Gregorian week})\)

Note that the same approach can be taken regardless of the calendar. In fact, multiple calendars can be used concurrently with the Date Ontology, simply by defining multiple such domain theories.

6.1.3 Other Approaches to Dates

A well-recognized approach to representing dates is found in the XML schema standard [3] with datatypes such as date, dateTime, gDay (i.e. “Gregorian day”), etcetera. This standard is based on ISO 8601 [47], the standard for representations of dates and times. Both standards are restricted to the Gregorian calendar, and while they take into consideration the properties of aggregation and repeatability in the definition of their parts, they are essentially syntactic restrictions on the format and thus offer little toward the goal of defining the semantics of dates.

An unfortunately common, overly simplistic approach to include the concept of dates in an ontology builds on these standards by leveraging the date datatypes as timestamp-valued labels for temporal concepts via a sort of 'has-date' property (we refer to this as the “datatype approach”). Not only are we unable to perform the type of reasoning described in the previous section with such a formalism, it also leads to an inaccurate representation of the key semantics of aggregation. This type of approach also fails to capture any of the semantics of the relationship between dates and time; rather the relationship is simply defined as a sort of label without any of the underlying meaning. In ontologies that conceptualized dates as time-identifiers in this way, it is possible to have situations where a time point represents the same date as an interval (e.g. January 22 and the interval represented as starting at the beginning of January 22 12:00 A.M. and ending at 11:59 P.M.). Similarly, it would be consistent for a time point (e.g. January 22) to contain one or many intervals (e.g. beginning at January 22 09:00 A.M. and ending at January 22 10:00 P.M.). Our understanding of time precludes us from including the aggregation of dates in this approach. When they are defined as specific time points it only makes sense to discuss a single level of granularity at once. These observations point to a need for an ontology of dates to formally characterise these properties such that they can be implemented in practice for more accurate reasoning and representation. This need has in fact been recognized previously in three attempts to formally axiomatize the semantics of the concept of dates: the Date-Time Vocabulary (DTV) [53], OWL-Time [45, 68], and Pat Hayes’ Calendars in [44].

The DTV has a concept of dates somewhat in line with the datatype approach, where a date is defined as a “time coordinate” (a name given to a point in time). However additional axioms account for the properties of aggregation and repeatability by reasoning about sequences and sets of sequences. OWL-Time represents aggregation and repeatability by identifying different types of relationships between temporal sequences of different types, opting for the use of second-order formulations in their axiomatization. The Calendars defined in [44] also quantify over sequences, where repeatability and aggregation are described with a “rhythm” that defines a sequence of durations that makes up a particular calendar unit of measure.

Prior to the development of the Date Ontology, no existing date ontologies accurately captured the semantics of the relationship between dates and time. This is likely a result of a focus only on modelling the behaviour of
dates that is, attempting to imitate the way in which dates unfold with respect to time, as opposed to considering the semantics of the concept. We speculate that this focus is also the cause of the awkward and unnatural form of these earlier axiomatizations. With such axiomatizations, implementing any of the reasoning tasks that served as motivation for an ontology of dates would likely be cumbersome and unable to scale well. Further, none of the existing axiomatizations have been proven to be consistent, nor have their models been characterized. The Date Ontology is an improvement over previous efforts as it is a straightforward first-order theory, and its models have been characterized up to isomorphism.

6.1.4 Dates via Reuse

The Date Ontology extends earlier work with durations \[33\] and introduces a first-order ontology that axiomatizes intuitions about dates and calendars. The fundamental ontological commitment is based on the observation that since dates have multiple repeatable occurrences at different timepoints, they can be defined as a class of activities. Calendars are complex activities composed of dates, and the axioms that specify how such complex activities occur are also introduced. Specific calendars, such as the Gregorian and lunar calendars, are domain theories that can be defined as extensions of the Date Ontology.

The Date Ontology consists of two modules (referred to as $T_{\text{date,periodic}}$ and $T_{\text{date.compose}}$) that formalize fundamental intuitions about dates. The remainder of this section will present the axiomatizations of these modules in more detail, along with a characterization of the models of the Date Ontology.

**Periodicity in the Date Ontology**

We first consider the module $T_{\text{date,periodic}}$, which axiomatizes the two intuitions about the periodicity of dates:

- Every date has multiple occurrences, each of which is associated with a pair of timepoints that are the beginning and end of the occurrence of the date (for example, each Monday begins at midnight and ends twenty-four hours later). Furthermore, all occurrences of the date have the same duration.
- Dates occur periodically, at fixed intervals along the timeline (for example, there is an occurrence of Monday in every week).

**Dates as Activities** Following the first intuition, dates appear to have the same properties as activities in the PSL Ontology \[30\]. We therefore axiomatize dates as classes of activities in an extension of the PSL Ontology.

Within the theory $T_{\text{pslcorn}}$ in the PSL Ontology, an activity is a repeatable pattern of behaviour, while an activity occurrence corresponds to a concrete instantiation of this pattern. The relationship between activities and activity occurrences is represented by the occurrence of $(o, a)$ relation. Activities may have multiple occurrences, or there may exist activities which never occur. Any activity occurrence corresponds to a unique activity.

Underlying the intuition that activity occurrences are the instantiations of activities is the notion that each activity occurrence is associated with unique timepoints that mark the begin and end of the occurrence. The PSL Ontology introduces two functions $\text{beginof}$ and $\text{endof}$ for this purpose.

Figure 6.1 depicts part of a structure which we want to axiomatize. We can see two dates, Monday and Tuesday, each of which has two occurrences; $o_{\text{today}}$ is the specific day which is an occurrence of Monday, which would make the next specific day $o_{\text{tomorrow}}$ to be an occurrence of Tuesday. There is another occurrence $o_{\text{next,Monday}}$ of Monday seven days later. Each day which is an occurrence of Monday or Tuesday has a

\[^2\text{http://colore.oor.net/psl_core/}^\]
timepoint at which the day begins and a timepoint at which the day ends. Notice also that a day is not an instance of a date, since Monday or Tuesday are both elements of the domain, rather than classes.

Structures for Date Periodicity The second basic intuition about dates is that they occur at discrete periods along the timeline. For example, in Figure 6.1, there are occurrences of the dates Monday and Tuesday every 7 days. In order to formalize this intuition, we also need to reuse an ontology for durations.

Duration Ontology The notion of duration plays a key role in the formalization of the second intuition about periodicity for dates. As with dates, there are several existing ontologies for durations that have been developed ([2], [44], [45]); these ontologies use a particular algebraic field (such as the rational or real numbers) to reason about duration. On the other hand, in the ontology for duration $T_{duration}$ proposed in [33] timedurations do not form a field, since we do not multiply timedurations, although we do want to multiply timedurations by a scalar (i.e. element of a field). Instead, timedurations form an ordered vector space $D$.

The ontology $T_{duration}$ also axiomatizes a duration function, which maps pairs of timepoints to a unique timeduration, and hence can be considered to be a vector-valued function. Within differential topology, a vector-valued function on the space $\mathbb{R}^n$ is known as a vector field [1], in which a unique vector is associated with each element of $\mathbb{R}^n$. However, since we want a mapping $\delta$ from $\mathcal{T} \times \mathcal{T}$ to the vector space $D$, we need to generalize the notion of vector field.

There are, of course, many possible functions from $\mathcal{T} \times \mathcal{T}$ to the vector space $D$. The class of vector-valued functions for duration is defined by using the automorphisms of the structure, that is, mappings from the structure to itself that preserve values of the duration function $\delta$:

$$\delta(t_1, t_2) = \delta(t_3, t_4) \iff \varphi(\delta(t_1, t_2)) = \varphi(\delta(t_3, t_4))$$

Intuitively, automorphisms of the timeline (which form a group denoted by $\text{Aut}(\mathcal{T})$) should preserve the duration function – if we shift timepoints along the timeline while preserving their ordering, the values of the duration function should not change. This insight leads to the following class of structures:

**Definition 29.** Let $\mathcal{T}$ be a timeline and let $\mathcal{D}$ be an ordered vector space.

The structure $\mathcal{V} = \langle \delta, \mathcal{T} \times \mathcal{T}, \mathcal{D} \rangle$ is a vector map iff

$\delta : \mathcal{T} \times \mathcal{T} \to \mathcal{D}$ and $\text{Aut}(\mathcal{V}) \cong (\text{Aut}(\mathcal{T}) \times \text{Aut}(\mathcal{T}))$

The following result from [33] characterizes the models of the Duration Ontology[^4] which we use in this paper:

[^3]: http://colore.oor.net/duration/
[^4]: The CLIF axiomatization of the theories referred to in the Theorem can be found at
Theorem 8. \( \mathcal{M} \in \text{Mod}(T_{\text{duration}}) \) iff
\[ \mathcal{M} = \langle T \cup D, \text{timepoint, before, timeduration, duration, add, mult, zero, one, lesser} \rangle \] such that:
1. \( T = \langle T, \text{timepoint, before} \rangle \in \text{Mod}(T_{\text{lp-ordering}}) \);
2. \( D = \langle D, \text{timeduration, duration, add, mult, zero, one, lesser} \rangle \in \text{Mod}(T_{\text{ordered, vector space}}) \);
3. \( \forall = \langle \text{duration, } T \times T, D \rangle \) is a vector map.

The vector space for timedurations and the vector map for the \( \text{duration} \) function in models of \( T_{\text{duration}} \) play a key role in the axiomatization of periodicity of dates.

Discrete Vector Maps \( T_{\text{duration}} \) allows us to say that two occurrences of a date correspond to a specific time-duration (e.g., there are 7 days between one Monday and the next). However, we still need to axiomatize the property that all occurrences of a date occur at discrete periods along the timeline. We therefore introduce the substructures of the timeduration vector space that are definable by occurrences of dates.

Definition 30. \( D' \) is semilattice in the vector space \( D \) iff it is a discrete additive subsemigroup of \( D \).

The intuition of “regular intervals” arises from the following property of semilattices:

Lemma 4. If \( D' \) is a semilattice in the vector space \( D \), then there exists a unique \( d \in D \) such that for any \( d_1, d_2 \in D' \), there exists \( x \) such that
\[ d_2 = \text{add}(d_1, \text{mult}(x, d)) \Leftrightarrow \langle d_1, d_2 \rangle \in \text{lesser} \]

We will refer to the element \( d \in D \) as the period of the semilattice \( D' \). In Figure 6.1, the period is the duration 7 days.

Although semilattices allow us to define sets of timepoints along the timeline, there are two sets of timepoints in which we are especially interested, namely, the timepoints at which occurrences of a date begin and end. We can use these timepoints to specify two special classes of substructures the vector map.

Definition 31. Suppose \( V = \langle \delta, T \times T, D \rangle \) is a vector map.
\[ V^b(a) = \langle \delta, T(a) \times T(a), D(a) \rangle \] is the substructure of \( V \) generated by the subset of timepoints
\[ T^b(a) = \{ t : \langle o, a \rangle \in \text{occurrence_of, } t = \text{beginof}(o) \} \]
and the subset of timedurations \( D^b(a) = \{ d : d = \delta(t_i, t_j), \ t_i, t_j \in T(a) \} \)
\[ V^e(a) = \langle \delta, T(a) \times T(a), D(a) \rangle \] is the substructure of \( V \) generated by the subset of timepoints
\[ T^e(a) = \{ t : \langle o, a \rangle \in \text{occurrence_of, } t = \text{endof}(o) \} \]
and the subset of timedurations \( D^e(a) = \{ d : d = \delta(t_i, t_j), \ t_i, t_j \in T(a) \} \)

In general, \( D^b(a) \) and \( D^e(a) \) will not be semilattices in the vector space \( D \), since it is not the case that all activities have periodic occurrences. Of course, this is precisely the class of activities which we are interested in for dates.

Definition 32. Suppose \( V = \langle \delta, T \times T, D \rangle \) is a vector map.
The vector map \( V(a) = \langle \delta, T \times T, D(a) \rangle \) is a discrete vector map iff \( D(a) \) is a semilattice in \( D \).

http://colore.oor.net/ordering/lp_ordering.clif and
http://colore.oor.net/algebra/vectorspace.clif
Using the notion of discrete vector maps, we can specify the required structures for the periodicity in the Date Ontology:

**Definition 33.** Let $\mathcal{M}^{date\text{-}periodic}$ be the following class of structures:

$\mathcal{M} \in \mathcal{M}^{date\text{-}periodic}$ iff $\mathcal{M}$ is the amalgamation of the structures $\mathcal{V}$ and $\mathcal{N}$ such that

1. $\mathcal{V}$ is a vector map;
2. $\mathcal{N} \in \text{Mod}(T_{pslcore})$;
3. if $\langle a \rangle \in \text{date}$ then $\mathcal{V}^b(a)$ and $\mathcal{V}^e(a)$ are discrete vector maps;
4. $\langle a, d \rangle \in \text{freq}$ iff $\mathcal{V}^b(a)$ and $\mathcal{V}^e(a)$ are discrete vector maps, and $d$ is the period of the semilattices $D^b(a)$ and $D^e(a)$.

Now that we have specified the required structures for this part of the Date Ontology, we turn to the problem of axiomatizing this class of structures.

**Axiomatization of Date Periodicity** The Date Ontology imports $T_{pslcore}$ (to axiomatize the distinction between dates and their occurrences) and $T_{duration}$ (to axiomatize timedurations and the duration function). The remaining axioms of the $T_{date\text{-}periodic}$ module within the Date Ontology can be found in Figure 6.2. Dates form a class of activities, each of which has a unique period (captured by the freq relation). Axiom 6.3 ensures that the set of occurrences of a date forms a discrete ordering. The period of an activity is the timeduration that is the value of the duration between successive occurrences of the date. By Axiom 6.4 all occurrences of a date have the same duration; together with the definition of frequency, this axiom guarantees that the set of $\text{beginof}$ timepoints for occurrences of the date generate a semilattice with the same period as the the semilattice generated by the set of $\text{endof}$ timepoints for occurrences of the date. Note that the axiomatization of $T_{date\text{-}periodic}$ depends on $T_{duration}$ to guarantee the existence of the timepoint at which the next occurrence of a date begins. The models of $T_{date\text{-}periodic}$ have the following representation theorem:

**Theorem 9.** The mapping

$$\mu : \text{Mod}(T_{date\text{-}periodic} \cup T_{duration} \cup T_{pslcore}) \rightarrow \mathcal{M}^{date\text{-}periodic}$$

is a bijection.

**Composition in the Date Ontology**

The third intuition about dates is that they can be composed into more complex dates that correspond to calendars and temporal aggregates. For example, each year is composed of months, each month is composed of days, and each week is composed of days. This also corresponds to the intuitions about temporal aggregates in [68], e.g. “five business days”, “every third Monday in 2014”, “every morning for the past four years”, “four consecutive Sundays”, “three weekdays after January 10”. In this section, we present the module $T_{date\text{-}compose}$ of the Date Ontology, which axiomatizes dates as a particular class of complex activities in the PSL Ontology. Since we are treating dates as activities, this means that we will need to consider a process ontology which axiomatizes complex activities.

**Complex Activities in the PSL Ontology** A useful feature of process formalisms is the ability to compose simpler activities to form new complex activities (or conversely, to decompose any complex activity into a set of subactivities). The PSL Ontology incorporates this idea while making several distinctions between different
\( (\forall a) \) date\((a) \supset (\exists d) \) freq\((a, d) \) 
\( (6.1) \)

\( (\forall a, d_1, d_2) \) freq\((a, d_1) \land freq\((a, d_2) \supset (d_1 = d_2) \) 
\( (6.2) \)

\( (\forall a, d, t, o_1, o_2) \) freq\((a, d) \land occurrence\_of\((o_1, a) \land occurrence\_of\((o_2, a) \land (d = duration(beginof(o_1), t)) \)
\( \supset \neg between(beginof(o_1), t, beginof(o_2)) \)
\( (6.3) \)

\( (\forall a, d, t, o_1, o_2) \) date\((a) \land occurrence\_of\((o_1, a) \land occurrence\_of\((o_2, a) \supset 
(duration(beginof(o_1), endof(o_1)) = 
duration(beginof(o_2), endof(o_2))) \)
\( (6.4) \)

\( (\forall a, t) \) occurs\((a, t) \equiv 
(\forall o) between(beginof(o), t, endof(o)) \supset occurrence\_of\((o, a) \)
\( (6.5) \)

\( (\forall a, d) \) freq\((a, d) \equiv (activity\((a) \land timeduration\((d) \land 
(\forall o) occurrence\_of\((o, a) \supset 
(\exists t) (d = duration(beginof(o), t) \land occurs(a, t))) \)
\( (6.6) \)

Figure 6.2: \( T_{date\_periodic} \): Axiomatization of dates as periodic activities. Additional axioms are imported from \( T_{pslc} \) and \( T_{dura} \).
kinds of composition that arise from the relationship between composition of activities and composition of activity occurrences.

The PSL Ontology uses the subactivity relation to capture the basic intuitions for the composition of activities. A model of the subtheory $T_{\text{subactivity}}$ is known as a subactivity ordered set, and it is equivalent to discrete partial ordering in which primitive activities are the minimal elements. This theory alone does not specify any relationship between the occurrence of an activity and occurrences of its subactivities.

Corresponding to the composition relation over activities, subactivity occurrence is the composition relation over activity occurrences. A model of the subtheory $T_{\text{actocc}}$ is known as a subactivity occurrence ordering, and it is a discrete partial ordering which is homomorphic to the subactivity ordered set. Occurrences of atomic activities are the minimal elements in this composition ordering – they do not have any nontrivial subactivity occurrences. One can consider the subactivity occurrence to be a temporal part of the complex activity occurrence. The axioms of $T_{\text{actocc}}$ guarantee that any subactivity occurrence is “during” an occurrence of the complex activity.

The basic structure that characterizes occurrences of complex activities within models of the subtheory $T_{\text{complex}}$ is the activity tree, which consists of all possible sequences of atomic subactivity occurrences. The relation $\text{root}(s, a)$ denotes that the subactivity occurrence $s$ is the root of an activity tree for $a$ and the relation $\text{leaf}(s, a)$ denotes that the subactivity occurrence $s$ is a leaf of an activity tree for $a$. Elements of the tree are ordered by the $\text{min\_precedes}$ relation; each branch of an activity tree is a linearly ordered set of occurrences of subactivities of the complex activity. The relation $\text{mono}(s_1, s_2, a)$ indicates that $s_1$ and $s_2$ are occurrences of the same subactivity on different branches of the activity tree for $a$. In addition, there is a one-to-one correspondence between occurrences of complex activities and branches of the associated activity trees. The set of activity trees in a model of $T_{\text{complex}}$ is known as a complex activity structure.

Figure 6.3: Structures for date composition of subactivities and subactivity occurrences.

**Structures for Date Composition** Since dates are defined to be a class of complex activities in the PSL Ontology, we first introduce the classes of substructures that are used to characterize the subactivities and activity trees for dates, as well as the substructures that characterize the subactivity occurrences of occurrences of a date.

**Definition 34.** $A^a$ is the substructure of the subactivity ordered set $A$ that is generated by the activity $a$.

$C^a$ is the set of activity trees for the activity $a$ in the complex activity structure $C$. 
C
 is the substructure of the complex activity occurrence ordering C generated by occurrences of the activity a.

Figure 6.3(a) is a depiction of A[gregorian_week], which is the substructure of A consisting of subactivities of the date gregorian_week. Figure 6.3(b) is a depiction of C[gregorian_week] for a specific occurrence of gregorian_week.

The axiomatization of the ontology is guided by more detailed intuitions for the composition of dates. First, observe that the days in a week, and the hours in day, are nested within each other. Further, the days in a week do not overlap each other, and similarly with the hours in a day. Thus, the occurrences of subactivities of a date correspond to intervals that are either disjoint or nested, and the resulting ordering is known as a series-parallel poset [9].

The second intuition about the composition of dates is that every subactivity of a date has a unique occurrence whenever the date occurs. For example, each occurrence of a week contains unique occurrences of the dates Sunday through Saturday. In terms of the structures within the models of the PSL Ontology, there is a bijection between the set of subactivities covered by a date in the subactivity ordered set A and the set of subactivity occurrences covered by an occurrence of the date in C. This intuition is made precise with the following notion:

**Definition 35.** A mapping \( \varphi : M \rightarrow N \) is a partial isomorphism iff there exist substructures \( M' \subseteq M \) and \( N' \subseteq N \) such that \( \varphi' : M' \subseteq M \rightarrow N' \) is an isomorphism, where \( \varphi' \) is the restriction of the mapping to the domain of \( M' \).

Combining these ideas leads to the specification of the required structures for date composition in the Date Ontology:

**Definition 36.** Let \( \mathcal{M}^{date\_compose} \) be the following class of structures: \( M \in \mathcal{M}^{date\_compose} \) iff \( M \) is an expansion of some \( N \in \mathcal{M}^{psl} \) such that for every \( (a) \in date \) we have

1. \( \omega : C^a \rightarrow A^a \) is a partial isomorphism, where \( (o, \omega(o)) \in occurrence; \)

2. \( C^a \cong \mathcal{P} \times \mathcal{K} \), where \( \mathcal{P} \) is a series-parallel poset and \( \mathcal{K} \) is an infinite antichain.

Condition (2) captures the additional intuition that all occurrences of a date have the same structure, that is, they are all occurrences of the same subactivities, and these subactivities always occur in the same linear ordering.

**Axiomatization of Date Composition** The axioms of \( T^{date\_compose} \) can be found in Figure 6.4. By Axiom 6.7 a date is a complex activity that is composed only of dates as subactivities, and by Axiom 6.11 the subactivity relation is isomorphic to a tree when restricted to dates. Axioms 6.8 and 6.9 guarantee that all occurrences of a calendar will have isomorphic activity trees, each of which consists of a unique branch. Axiom 6.10 corresponds to the property that the subactivity_occurrence relation is isomorphic to a tree when restricted to occurrences of dates.

The models of \( T^{date\_compose} \) have the following representation theorem:

**Theorem 10.** The mapping

\[ \mu : Mod(T^{date\_compose} \cup T^{psl}) \rightarrow \mathcal{M}^{date\_compose} \]

is a bijection.
\[(\forall a_1, a_2) \text{ date}(a_1) \land \text{subactivity}(a_2, a_1) \supset \text{date}(a_2)\]  \hspace{1cm} (6.7)

\[(\forall a_1, a_2, o_2) \text{ date}(a_1) \land \text{subactivity}(a_2, a_1) \land \text{occurrence}_o(a_1, a_1) \supset (\exists o_2) \text{occurrence}_o(a_2, a_2) \land \text{subactivity}_o(a_2, a_1)\]  \hspace{1cm} (6.8)

\[(\forall a, s_1, s_2, s_3, s_4) \text{ date}(a) \land \text{min_precedes}(s_1, s_2, a) \land \text{mono}(s_1, s_3, a) \land \text{mono}(s_2, s_4, a) \supset \text{min_precedes}(s_3, s_4, a)\]  \hspace{1cm} (6.9)

\[(\forall a_1, a_2, o_1, o_2, o_3) \text{ date}(a_1) \land \text{occurrence}(o_1, a_1) \land \text{subactivity}_o(o_2, o_1) \land \text{subactivity}_o(o_3, o_1) \land \text{occurrence}(o_1, a_1) \land \text{occurrence}(o_3, a_1) \land \text{occurrence}(o_2, a_2) \land \text{occurrence}(o_3, a_2) \land (o_2 \neq o_3) \land (a_3 \neq a_1) \land (a_3 \neq a_2) \land \text{min_precedes}(s_1, s_2, a) \land \text{min_precedes}(s_2, s_3, a) \supset \text{min_precedes}(s_4, s_3, a)\]  \hspace{1cm} (6.10)

\[(\forall a, a_1, a_2, s, s_1, s_2, s_3, s_4) \text{ date}(a) \land \text{subactivity}(a_1, a) \land \text{subactivity}(a_2, a) \land \text{root}(s_1, a_1) \land \text{leaf}(s_3, a_1) \land \text{root}(s_2, a_2) \land \text{root}(s_4, a_2) \land \text{min_precedes}(s_1, s_2, a) \land \text{min_precedes}(s_2, s_3, a) \supset \text{min_precedes}(s_4, s_3, a)\]  \hspace{1cm} (6.11)

---

Figure 6.4: $T_{date\text{-compose}}$: Axiomatization of date composition. Additional axioms are imported from $T_{actocc}$, $T_{complex}$ and $T_{subactivity}$. 
CHAPTER 6. EXAMINING THE BENEFITS OF REUSE IN PRACTICE

6.1.5 Summary

Rather than becoming entangled in the more direct approach of attempting to model the semantics and behaviour of dates from-scratch, the Date Ontology leverages a similarity between the intuitions of dates and those of activities. This allows for an approach to the problem of a representation of dates through the reuse of existing theories, resulting in an effective, clean axiomatization with only eleven additional axioms. The advantages extend beyond reducing the necessary development effort. The ontologies were reused by conservative extension (instances of domain-combination and domain-extension, according to the framework presented in the previous Chapter), therefore their semantics are preserved in the Date Ontology. It is straightforward to see that shareability is readily supported with any applications that implement the PSL ontology, which could be advantageous for the development of scheduling applications for example. Further, an additional benefit of reuse arises for the verification of the Date Ontology. Previous results were leveraged to develop characterizations of the required classes of structures, and representation theorem proofs for the models of the axioms.

6.2 A Home Services Ontology

The Home Services Ontology (HSO) was developed to provide proof-of-concept for the use of an ontology to support the front-end functionality of the website Servicedathome.com through semantics. Servicedathome.com is an online service developed by Hunch Manifest Inc. to assist users with home improvement projects by identifying the required services and resources, as well as to provide recommendations for potential service or resource providers. The HSO formally defines the concepts in the domain for two purposes: (1) to avoid clashes in the intended meanings of users’ project descriptions and inquiries, and (2) to support the use of automated reasoning to provide answers to the users’ inquiries.

As illustrated in Figure 6.5, the Home Services Ontology (HSO) was built as an extension of gist (version 6.7)\(^5\) and PSL.owl\(^6\) – a sub-theory of the PSL ontology \(^3\) that has been developed in OWL. While gist is meant as a minimal upper-level ontology, and therefore has a relatively broad scope of concepts, PSL is more activity-oriented, defining the concepts related to activities and their occurrences in more detail than is provided in gist. The related concepts between the two ontologies are integrated to provide the required breadth and depth for the HSO.

6.2.1 Modifications to gist

The development of the HSO required modifications to gist to avoid inconsistencies when it was combined with PSL.owl. The original ontologies, gist and PSL.owl are in fact inconsistent when combined with the bridge axioms in the HSO. The HSO conceptualizes the Event class to be equivalent to PSL.owl’s Activity_Occurrence class and similarly the Behaviour class to be equivalent to PSL.owl’s Activity class. In gist, the Event class is defined as a TimeInterval characterizedAs some Behavior. However, the definition of the SpeechAct class in gist is inconsistent with this conceptualization; the SpeechAct class is defined as a subclass of both the Behavior and Event classes, whereas the Activity and Activity_Occurrence classes are disjoint in PSL.

To address this, the HSO reuses gist via a weakening extraction: the axiom that defines SpeechAct as a subclass of Behavior is removed in the version of gist reused for the HSO. The choice to modify gist to resolve this reflects the ontological commitment of the HSO that activities and their occurrences are disjoint, as in PSL.owl.

\(^5\)http://semanticarts.com/gist/
\(^6\)http://stl.mie.utoronto.ca/ontologies/processspecificationlanguage/psl_loc_actors.owl
6.2.2 Summary

The HSO demonstrates a reduction of workload made possible via the reuse of gist - which provides a good breadth of concepts, and PSL.owl - which provides a relatively deep axiomatization of activities. The weaker version of gist that is reused is strengthened by the additional combination with PSL.owl and strengthening extensions in the HSO (i.e. the bridge axioms). However, it is straightforward to see that the resulting HSO is in fact incomparable to the original version of gist that was reused. While the inconsistency may have resulted from only a single axiom, it nevertheless represents a barrier for shareability with gist and its other applications. PSL.owl on the other hand, is reused via strengthening domain extension and combination. The resulting HSO is a non-conservative extension of PSL.owl, and thus not shareable with other applications of PSL. However, if the HSO had created new IRI’s for the extensions of the PSL concepts as discussed in the previous chapter, it could have preserved the original semantics of PSL.owl in the HSO and attained the benefit of shareability at least with other implementations of PSL.owl.

6.3 Factors in the Benefits of Reuse

In the previous chapter we discussed the implications of the different reuse operations on the benefits of reuse, in particular the impact on shareability. The two example ontologies described here demonstrate how the developers’ requirements factor into the benefits of reuse. Specifically: (1) the ontological commitment of the developers, as this determines whether certain extractions and / or extensions must be made to capture the intended models, and (2) the selected / required ontology(s) for reuse, as their content and the degree with which it agrees / disagrees with the ontological commitments will dictate not only how much additional development is required, but whether any shareability will be possible with the resulting ontology.

The definitions provided in Chapter 5 aid in the tractability of benefits by providing the developer with tools to look-ahead to the consequences of reuse of a particular candidate, prior to investing the necessary time and
resources into the design. This provides a means of concretely assessing the impact of a particular ontological commitment or the choice of a particular candidate.
Chapter 7

An Infrastructure for Reuse

In this chapter we consider how the solutions of this thesis can be operationalized. In the previous chapters, we developed solutions to address both of the objectives of this thesis: to minimize the developers’ reuse-related workload, and to achieve consistent, predictable realization of the benefits of reuse. However, this content alone is insufficient to achieve the vision for reuse. In part this is because there is still a considerable distance between the solution, identified ‘on paper’, and the solution in operation. A means of supporting the collective adoption of these solutions by both the individual developer and the ontology community is required; for example, to account for certain assumptions that the solutions make that are not guaranteed in the current state. The final piece remaining in order for the solution to be viable is a supporting infrastructure. The infrastructure would support and coordinate all of the solutions presented here as they apply to the various tasks of reuse. It could be comprised of automated functions, human-supported functions, or both. In any case, the infrastructure allows the developer to be concerned solely with the tasks of development via reuse and not the details of the solutions presented here.

In this chapter, we define a standard for such an infrastructure. The requirements for this standard are derived from the solutions presented in the previous chapters. Thus, we guarantee that any conforming infrastructure shall fulfil the objectives of the thesis in practice.

7.1 Scope

This standard is relevant for any community or group that wishes to see an increase in the occurrence of reuse; it is applicable for all organizations and usual ontology languages (e.g. RDF, OWL, FOL). The scope of the solution presented here includes only those instances of the reuse process that satisfy the definition of reuse and reusability presented in the previous chapter; claims regarding the objectives of reduced work and increased benefits apply only to such instances of reuse. Further, the minimization of workload achieved is refers to a minimization of the work that developer is responsible for performing; it does not extend to consider minimizing the sort of cognitive workload that is the focus of other disciplines, such as the effort required to perform an individual task that could be increased or reduced as a function of some interface design. This is an important consideration in its own right, however it is one that is better left to experts in those fields.

The infrastructure consists of a Search Component and a Choice Component, as illustrated in the high-level diagram in Figure 7.1. Each component supports a task in the process of design via reuse, as described in Chapter 3.
A natural recommendation to simplify the process of reuse is with the introduction of supporting tools and technology. This is one of several factors that we attribute to the relative success of reuse in the Semantic Web community, as we observe a correlation between the availability of infrastructure and tool support (via the Semantic Web, Protege\textsuperscript{1} and a variety of commercial tools) and the adoption of design via reuse. While not necessary for the guarantees we make here and thus outside outside of the scope of the standard, tool support certainly has the potential to encourage reuse and the uptake of these solutions; it would make a valuable addition toward the goal of achieving the vision of reuse. Therefore, while it is not a formal requirement of the infrastructure, it is advisable that the functions provided be allocated to automated solutions as much as is practical and feasible for its application.

### 7.2 Conformance Requirements

The infrastructure shall support the division of labour described in Chapter\textsuperscript{5}, the means of attaining the benefits of reuse described in Chapter\textsuperscript{5} as well as any incidental tasks that are indirectly required, such as supporting the workflow connection between the developer and the independent resources that are responsible for the objective tasks.

\textsuperscript{1}http://protege.stanford.edu/
7.2.1 Search

In Chapter 3 it was determined that all of the tasks of search could be offloaded from the developer due to their objective nature. Thus, the Search Component of the infrastructure must be responsible for coordinating and executing these tasks. This component interprets the specified requirements as Search Criteria, and applies them to a source(s) of ontologies in order to retrieve the relevant candidates. As illustrated in Figure 7.2, this requires a tool (human or automated) to implement the search procedures described in Chapter 3, as well as an accessible source of ontologies.

Figure 7.2: The Search Component of the infrastructure.

R1 The Search Facility shall provide tool support to implement a procedure for the retrieval of candidates and associated conjectures of relevance based on the semantic requirements.

As discussed in Chapter 3, there exists a range of approaches to the derivation of relevance conjectures from a set of semantic requirements. The appropriateness of any given approach is dictated by the efficiency of the tools available, but also by factors such as the size of the search space, and perhaps the domain of application itself. For example, if it is known that there is little to no vocabulary variation in a given domain (legal or standards, for example) then the search need not extend far beyond basic keyword matching. However regardless of the situation, the goals of high precision and recall remain and the approach to search should reflect this aim.

R2 The Search Facility should implement an approach to generating conjecture criteria that is as thorough as possible or reasonable.

In Chapter 2, we saw there are not only a number of sources of ontologies, but also a variety distinct types of sources (such as repositories and search engines). To ensure a thorough search in which an effort has been made to identify and consider all possible benefits of reuse, in theory the search should include all available sources of ontologies. However this can easily become impractical and what’s more, whether the task is charged to a human or a machine, how is the infrastructure to ensure no sources have been omitted? Regardless of who or what executes the retrieval of candidates, in order for this task to be performed reliably and effectively there must be some structure in the sources of ontologies. Retrieval would be most straightforward if there were a single, standard source of ontologies, or (more realistically) a single point of access to multiple ontology sources. Alternatively, a record of the current sources and their access protocol could be maintained, however this approach is less scalable/flexible and thus not preferred.
R3 The infrastructure shall have either a single source of ontologies, or a single point of access to search a collection of ontology sources.

Finally, with regards to specific types of sources: in general, the approach taken to organize and store ontologies has no bearing on the ease of search so long as the other requirements are satisfied. One exception exists in form of upper ontologies; their approach as a source for ontologies is fundamentally distinct as they typically do not allow for multiple ontological stances. Due to this, they are not so much a source for search, but represent a particular candidate. In fact, previous work [38] showed how upper ontologies could be more effectively stored and represented in a logically-structured ontology repository. Based on this analysis we find that upper ontologies alone are not adequate sources of ontologies for reuse and that, if included, they would be better treated as potential candidates.

R4 The infrastructure shall not provide only an upper ontology as a source for ontologies.

R5 If applicable, the infrastructure shall store and treat upper ontologies as it does the other ontologies in its source(s).

7.2.2 Choice

The final decision, while subjective and fully in the hands of the developer, is supported in various ways by the Choice component as illustrated in Figure 7.3. According to the division of labour identified for the task of Choice, the infrastructure shall support the evaluation of the semantic and objective requirements. In addition, the results of Chapter 5 must be supported here. The transparency of the benefits of reuse is crucial to the user as a factor in the informed decision of which ontology to reuse, if any.

In Chapter 3 we identified the offloading of objective, non-functional requirements as one requirement for the minimization of the developer’s workload.

R6 The Choice Facility shall evaluate all candidate ontologies against any objective NFRs specified in the Requirements.
The true objectivity of such requirements, and the ability to offload them relies on the availability of required metadata, and a standardized interpretation of the requirements. If both of these conditions are not met then the assessment may vary, and we cannot guarantee its objectivity. We can ensure these conditions are met for some standard set of NFRs; via a formal definition of the NFRs that identifies a standard interpretation (i.e. means of evaluation), and via provision of the necessary metadata. It is impossible to specify a complete set of NFRs, objective or otherwise. However, these conditions can be guaranteed for some standard subset of NFRs; this set of NFRs will be clearly defined and will enable a consistent, objective evaluation and minimized workload.

R7  The infrastructure shall define a standard set of objective NFRs for reuse.

R8  The infrastructure shall only permit sharing of ontologies with the standardized metadata necessary to assess each objective NFR.

Since the semantic requirements are completely objective, the division of labour prescribed in Chapter 3 also required that their evaluation be offloaded from the developer.

R9  The Choice Facility shall evaluate the candidate ontologies against all specified semantic requirements.

R10 The Choice Facility shall provide the developer with the results of each theory’s evaluation results.

The final step in the task of Choice requires the developer to use the results of the requirements’ evaluation of each candidate as input to determine which, if any, candidate ontology to reuse. While the final decision is subjective, as discussed in Chapters 3 and 5 there are some key assessments that are in fact completely objective. These should also be offloaded; their results and the results of the requirements’ evaluation should then be communicated to the developer to support the final decision.

R11 The Choice Facility shall implement a procedure (such as defined in Chapter 4) to obtain an partial ordering of all candidates retrieved from search, based on the notion of Preference with respect to the semantic requirements.

The effectiveness of this procedure relies on certain assumptions. To apply the procedure, the candidates must be both accessible and machine-readable. We expect and account specifically for CommonLogic or OWL/XML formalizations in this work, however other representations are possible so long as a translation to some standardized format exists.

R12 All ontologies in the infrastructure shall be shared and accessible in a known repository. They must be available in a standardized, machine-readable form.

The procedure also assumes the theories are consistent and modularized. The theories’ modular structure is also key in the analyses presented in Chapter 5. As discussed, model-generation tools may be employed to check consistency, and a procedure to modularize ontologies exists. While these steps could be included as part of the procedure, it would be more practical (and advantageous to the community overall) if these were implemented as a “gatekeeping” processes. Unfortunately in the current state, at best we find ontologies are modularized post-design, whereas many are never modularized at all.

R13 All ontologies in the infrastructure shall be consistent and modularized.

As we saw in Chapter 5, there are dimensions beyond the CQ results that characterise the task of reuse. The evaluation facility will must also consider these dimensions in its evaluation of the competency questions, as...
illustrated in Figure 3.3. In Chapter 5, we showed how the definition of reuse could be applied to provide guidance for the design procedure by identifying the necessary reuse operations to achieve the required theory from a particular candidate. These results also serve as useful input for the task of choice, thus they should be calculated and provided to the developer as an additional assessment.

R14  The Choice Facility shall implement the approach (defined in Chapter 5) to identify the necessary reuse operations for each candidate theory.

Chapter 5 also considered how the different reuse operations impacted the promise of shareability between theories. The necessity for improved understanding of this outcome was discussed, and we identified an approach to determine whether shareability would be attained between the required ontology and a given candidate theory. For transparency and control over this benefit, it was determined that this assessment also factors into the final choice.

R15  The Choice Facility shall implement an approach (described in Chapter 5) to identify the necessary reuse operations, and their implications on the preservation of a candidate’s semantics.

Finally, it is unavoidable that the way in which the infrastructure’s interface is implemented will increase or decrease its impact on the simplification of this task. The specifics of interface design are out of the scope of this standard, thus conformance is not contingent on any particular interface design; the focus of this specification is on what must be provided rather than how it should be presented. However, given the potential impact of a good or bad design, any attempt to satisfy this standard should make some effort to ensure the quality of its interface with the designer.

R16  The infrastructure’s design shall demonstrate at minimum some consideration of basic usability/human factors design principles.

7.3 Supplementary Recommendations

The conformance requirements presented above define the bare minimum required for an infrastructure to ensure satisfaction of the objectives set out in this thesis. In an effort to implement a high quality solution, it is also important to consider what additional factors would be useful augmentations in the creation of such an infrastructure. These requirements are not strictly necessary to achieve the objectives of the thesis, however they have the potential to improve the infrastructure and simplify its implementation. This section provides a review of these supplementary recommendations.

If the infrastructure includes a number and variety of sources of ontologies, there is a potential for redundancy in the ontologies provided. More seriously, this introduces the risk of confusing newer and older versions of the same ontology. While versioning remains an open issue in the ontology community, to avoid this the infrastructure should leverage IRIs and other metadata such as version number or date of creation.

R17  If there is more than one source of ontologies, it is advisable that the search mechanism employed by the infrastructure rely on IRIs or similar technology to resolve duplicate theories.

While the solutions presented in this work do not rely on the use of a repository, we have seen how the relationships provided by a repository such as COLORE would be advantageous for tasks in both Search and Choice.
R18 It is advisable that repositories be used as the infrastructure’s source(s) for candidate ontologies. While the procedures themselves do not require the representation of relationships between theories, the ability to store such information is useful from a practical standpoint to avoid the need for redundant calculations. This is particularly important as the procedures are not trivial. For this reason we require the infrastructure to perform and store these tasks with a mechanism that supports the relationships between theories used here; i.e. hierarchies, conservative extensions, and non-conservative extensions. This will support storage of results of all of the CQ-based evaluations, further if this is implemented in the ontology source itself it will also provide a structure that will allow past results to be leveraged intelligently for Search (conjecture generation) heuristics.

R19 It is advisable that the source(s) used in the infrastructure support the definition of the following relationships between theories: hierarchies, conservative extensions, non-conservative extensions.

Finally, regarding the recommendation for NFR evaluation: in general, no standard guideline or set of NFRs exist. In the literature we find only general suggestions and examples as in the Ontology Requirements Specification Document (ORSD) [79], or definitions of subjective qualities as in [29]. However, there do exist standards for ontology metadata - two efforts in particular are well-recognized: the Ontology Metadata Vocabulary (OMV) [43], and the Distributed Ontology Modeling and Specification Language (DOL)[46]. Rather than attempt to develop and recommend a standard set of NFRs, a well-established standard of metadata may be used starting point for such a set. A set of metadata developed as part of a standard is advantageous as it has achieved collective agreement regarding its importance, indicating a general opinion that there is useful insight to be gained from it. A set of objective NFRs may then be explicitly defined with respect to this metadata; this may be done uniquely for each development project, or a priori for all development projects.

OMV was a product of the NeON project[2], the same group that developed the ORSD which also contained a functional (CQ) and non-functional specification of requirements. In fact, one of the objectives of the OMV was to facilitate reuse; past documentation indicates it was well-aligned with the goal of NFR evaluation that we have here. Unfortunately it appears that the OMV effort is no longer active, and the reason for this is unclear.

DOL was already encountered in Chapter[5] as we identified relationships between its Structuring Language and the reuse operations defined in this thesis. There, we recognized that the goal of DOL, “to achieve integration and interoperability of ontologies, specifications and MDE [Model-Driven Engineering] models” resulted in a distinctly different perspective than the one taken here. While we were able to identify relationships between each of the reuse operations and the DOL vocabulary, in some cases DOL was not detailed enough to capture certain distinctions of interest. Conversely, due to its broader scope of focus, there were many DOL terms that were not related to any of the reuse operations.

While the motivation of the OMV may be more in-line with the work presented here, the project’s apparent inactivity is a major disadvantage. This means it may be out of date or have unresolved issues, and there will be little motivation for positive feedback between the metadata standard and the infrastructure’s standard NFRs. DOL on the other hand is an active effort, however its metadata will in some cases be broader than what may be applicable here. The ideal metadata standard would be some combination of the two standards, thus rather than attempt to resurrect the OMV, we propose the use of a modified subset of DOL.

R20 It is advisable that the infrastructure enforce the use of a subset of DOL that includes only concepts relevant to ontologies. Further, this infrastructure should include additional, reuse specific metadata such as the reuse operations identified in Chapter[5].

---

A further advantage to the use of DOL is that it is in the final stages of the review process to become an OMG Standard. We can therefore be confident in both its correctness and quality, and expect that this information should become more readily available as the standard is adopted. Any NFRs that are specified outside of the standard set represent useful feedback for the chosen standards (e.g. DOL); as we cannot make any guarantees regarding the completeness of any NFR set, it should be expected that required modifications or additions will be identified over time to maintain and improve its usefulness in practice. For example, the contextual metadata discussed in [8], though not required, may be a useful addition to the standard set of metadata.

It is advisable that the infrastructure evolve the standardized NFRs and metadata at regular intervals, based on feedback in the form of usage summaries or community discussion.

7.4 Integration with Existing Solutions for Ontology Development

The set of requirements presented here considers the fundamental infrastructure required to support the solutions for reuse presented. While the infrastructure relies on the support of existing tools such as sources of ontologies, the solutions’ implementation is considered independently of any particular tool(s); this was necessary in order to remain tool-agnostic. We conclude by considering how the work presented in this thesis could be implemented with current solutions for ontology development. The hope is to make the preceding infrastructure description more concrete, and perhaps even to motivate the adoption of these solutions by existing tools and techniques.

Existing approaches to ontology development, such as METHONTOMETRY [18], and our own earlier work presented in [50, 49] could easily be integrated with the solutions for reuse presented here. As recognized in Chapter 3, reuse is often recognized as a potential task in ontology development – in fact, it is identified as a step in both of the referenced methodologies. Therefore integration becomes a matter of simply substituting formal, rigorous techniques described here with the relatively high-level descriptions in existing work. Further, given the observations made in Chapter 3, this straightforward integration applies to any traditionally structured approach to ontology development, regardless of whether a Reuse step has been identified. We observed that the process of Reuse is a special case of design; consequently, all of the solutions developed here may simply be implemented as part of the Design stage of development – assuming the use of CQs are part of the Requirements Specification.

At the beginning of this Chapter, we discussed the value of tool support for the solutions presented here. The integration with existing tools for ontology development is an important consideration as it presents an attractive alternative to the design of a completely new tool(s) to support the infrastructure prescribed here. Apart from some special purpose tools that have been developed for tasks such as error detection [80, 74], to-date the closest thing to tool support for ontology development may be found in existing ontology design tools. Currently, two of the most popular such tools are Protegé[3] and TopBraid Composer (TBC)[4]. Protegé is a freely available tool, with an established history in the ontology community that was discussed earlier in the thesis. TBC is a relatively newer, commercially available tool that has gained popularity owing to its wide range of implementation-oriented capabilities. While the tools support the implementation of ontologies to varying degrees, their support for the design of axioms is similar. They provide a graphical user interface for users to define and modify concepts in order to create their ontologies. In the context of such tools, the solutions presented in this thesis, and the required infrastructure itself can essentially be seen as enhancements to the ‘import’ functionality offered by both tools.

Currently, the import paradigm implemented by these design tools amounts to a very basic function of specifying an ontology to load into the development project. This is the extent of the current support for design via reuse in existing tools, however we have seen that the task of reuse extends well beyond this. Current import functionality could be augmented with the Search and Choice components of the infrastructure described here. This would mean that existing tools would need to extend their capabilities to support the specification of requirements, as well as the integration with sources of ontologies. If only for addressing the challenges of candidate search and selection, this would be a worthy endeavour.

Beyond this, the identification of necessary operations for reuse as discussed in Chapter 5 could be implemented to augment the design functionality supported by these tools and provide guidance for the development of imported theories. The notion of assessing shareability based on the necessary specialized operations could also be leveraged to automatically generate metadata regarding the relationship between the ontology being designed and the reused ontologies. This information could be further extended not just to inform users, but to actively address the current issues with reuse identified in the same chapter. For example, a design tool such as Protégé might be triggered to automatically create a new IRI for a term from an imported ontology as soon as axioms are added that result in a non-conservative extension.

The work required to extend these existing tools is certainly not trivial, nor does it cover all instances of reuse (since these tools support Semantic Web ontologies, the above extensions would not benefit the development of FOL ontologies, nor the extended notion of reuse presented in the subsequent chapter). However, this is a direction that is worth considering as it offers the ability to heighten the impact of the solutions presented here by making them readily accessible to users precisely when they are required.
Chapter 8

Reuse with Translations

In the previous chapters, we presented a set of solutions to achieve the objectives for reuse. The definitions of reuse and reusability, as well as some of the techniques that comprise these solutions assume that no signature translations are required, and that no logical language translations are required for reuse. These assumptions are typical of current practice, however we recognize that it would be prudent for the definition of reuse to be extended to consider less conventional candidates, specifically those requiring signature or logic translations.

In this chapter, we consider a broader definition of reuse that includes the possibility of translations between signatures and logical languages. We reflect on the implications this definition will have on the solutions that we have developed in the previous chapters. We have already shown how these solutions will achieve our objectives in the current state; the aim of this chapter is to reflect on the impact of an extended definition of reuse and consider whether the solutions will still be effective should this extended definition become a regular practice of reuse.

We motivate the potential benefit of such an extended definition of reuse by providing two examples of cases in which reuse with translations was applied with positive results. Each of these examples is derived from previous work [39, 52]; this work is summarized here for brevity, however more detailed presentations are supplied in Appendix C. Each example demonstrates a different type of translation; we then provide a precise definition of the concepts of signature and logic translations. Following this, we formalize the notion of reuse-with-translations and consider the implications of such a definition on the solutions for reuse that we have presented.

8.1 Translations Defined

Before extending the definition of reuse to include translations, it is important to identify the notion of translations more precisely. In the context of reuse, we are concerned with two types: signature translations and logic translations.

8.1.1 Signature Translations

Informally, signature translations allow us to consider the semantics of a theory for different terminology in the same, or possible completely different domains.

Essentially, in a signature translation we map the concepts (signature) from one theory into the concepts of another. This is closely aligned to the definition of an interpretation from a language into a theory as specified in [15]. The key difference in our notion of signature translations is that we do not require them to be complete
(whereas the language interpretation in [15] specifies a mapping on the whole language). For our purposes the mapping of even a single term in a language is of interest, thus a signature translation is defined as an extended notion of a language interpretation.

**Definition 37.** Given some theory, $T_1$, in signature $\sigma(T_1)$ and some theory $T_2$ in a (possibly partially or completely different) signature $\sigma(T_2)$, a signature translation of $T_1$ into $\sigma(T_2)$ is an interpretation $\pi$ of $T_1$ into a subtheory of $T_2$, $T'_2 \subseteq T_2$, as defined in [15]: An interpretation $\pi$ of the theory $T_1$ with signature $\sigma(T_1)$ into a theory $T'_2$ with signature $\sigma(T'_2)$ is a function on the set of non-logical symbols of $\sigma(T_1)$ and formulae in $L(T_1)$ such that

1. $\pi$ assigns to $\forall$ a formula $\pi_v$ of $\sigma(T'_2)$ in which at most the variable $v_1$ occurs free, such that $T'_2 \models (\exists v_1) \pi_v$

2. $\pi$ assigns to each $n$-place relation symbol $P$ a formula $\pi_P$ of $\sigma(T'_2)$ in which at most the variables $v_1,...,v_n$ occur free.

3. for any atomic sentence $\Sigma \in \sigma(T_1)$ with relation symbol $P$, $\pi(\Sigma) = \pi(P)$;

4. for any sentence $\Sigma \in \sigma(T_1)$, $\pi(\neg \Sigma) = \neg \pi(\Sigma)$;

5. for any sentences $\Sigma, \tau \in \sigma(T_1)$, $\pi(\Sigma \supset \tau) = \pi(\Sigma) \supset \pi(\tau)$;

6. for any sentence $\Sigma \in \sigma(T_1)$, $\pi(\forall x \Sigma) = \forall x \pi_v \supset \pi(\Sigma)$;

7. For any sentence $\Sigma \in \sigma(T_1)$, $T_1 \models \Sigma \Rightarrow T'_2 \models \pi(\Sigma)$

**A Motivating Example: Identifying Reuse of Ontology Patterns**

Leveraging the formal relationships provided by the ontology repository COLORE, examples can be found to illustrate how COLORE (and formal repositories in general) may be aligned with the idea of ontology design patterns (ODPs). We are able to show how a variety of real-world ontologies are in fact extensions of the same group of theories, thus these generic ontologies are in fact playing the role of ontology patterns. Theories from the same hierarchy are reused in ontologies for parthood, periods of time, as well as subactivities. Since these theories are axiomatized with a relatively abstract signature, they would not have corresponded to the signatures of any of the required theories’ during development. However, the interpretations of the axioms were such that they could be reused to define the semantics required for this variety of domains. By considering reuse with signature translations, a seemingly unrelated theory may in fact provide precisely the required semantics that are required. This approach greatly expands the pool of potential candidates, thus encouraging reuse and increasing the possibility of finding an appropriate ontology.
8.1.2 Logic Translations

The concept of translations between logical languages has been addressed in previous work, such as [58, 56]. The notion of logic translations we consider here is similar, but we expand our focus to consider less-traditional translations, including those that may not be provably complete. Translations between logics may be defined as different comorphisms as in [25] and applied in [56]. We adopt the formalization from [56] where a comorphism from a logic $L_1$ to a logic $L_2$ consists of:

- a mapping $\Phi$ of $L_1$-signatures to $L_2$-signatures,
- for each signature $\sigma$, a mapping $\alpha_\sigma$ of $\sigma$-sentences in $L_1$ to $\Phi(\sigma)$-sentences in $L_2$, and
- for each signature $\sigma$, a mapping $\beta_\sigma$ of $\Phi(\sigma)$-models in $L_2$ to $\sigma$-models in $L_1$,

such that:

$$\beta_\sigma(M)^{L_1} \models \varphi \text{ iff } M \models^{L_2} \alpha_\sigma(\varphi)$$

for each signature $\sigma$, $\sigma$-sentence $\varphi$, and $\Phi(\sigma)$-model $M$.

It is a faithful comorphism if logical consequence is preserved. In other words, for any theory $T$ and sentence $\varphi$ in $L_1$:

$$T \models^{L_1} \varphi \text{ iff } \alpha(T) \models^{L_2} \alpha(\varphi)$$

The concept of a logic translation builds on this notion of a faithful comorphism to consider for translations between theories from more- to less-expressive logical languages (e.g. FOL to OWL). In such cases due to limitations of the language a faithful comorphism between the logics does not exist. Thus, we define a logic translation as follows:

**Definition 38.** Given 2 theories, $T_1$ in logic $L_1$ and $T_2$ in logic $L_2$, $T_2$ is a logic translation of $T_1$ iff:

- there is a faithful comorphism from $L_1$ to $L_2$ and the comorphism mapping on the sentences in $T_1$ is logically equivalent to $T_2$.
  $$\alpha_{\sigma(T_1)} = T_2, \text{ or}$$
- there is a faithful comorphism from $L_2$ to $L_1$ and not from $L_1$ to $L_2$ and the comorphism mapping on the sentences in $T_2$ are logically equivalent to a subtheory of $T_1$.
  $$\alpha_{\sigma(T_2)} = T'_1 \quad T'_1 \subseteq T_1$$

Using PSL to Extend Event Ontologies

Currently, we find there exist a variety of efforts toward the representation of events for the Semantic Web. Although contrary to the principles of reuse, this is not entirely surprising given that the concept of an event plays an important role in such a wide range of contexts. A review of existing event ontologies agrees with this observation as we have found the application areas for existing work to be quite diverse, including domains such as artefacts, historic events, and business processes. The focus of most existing efforts is on information integration, and while this is certainly a valuable application, more complex reasoning can and should be performed with the event information on the web.

Given the strides that have been made for integration and basic information retrieval for event-related information, it is logical to now ask what can be done with this information. Many existing event ontologies tend to take...
a rather simplistic approach in order to better facilitate use with many diverse sources, resulting in a rather limited semantics. To remedy this, we attempted to strengthen these existing ontologies by reusing them in combination with a more rigorous axiomatization. PSL, an ISO standard that provides a rich axiomatization for processes was a natural choice to strengthen the Semantic Web ontologies’ semantics; however PSL is formalized in FOL, whereas the Semantic Web ontologies are formalized in OWL. Since the goal was to increase the reasoning abilities of these ontologies on the web, the resulting ontology would need to be formalized in OWL. Therefore, an OWL version of PSL was created to enable the reuse of PSL.owl via combination with the Semantic Web event ontologies. Subsequent evaluation of the resulting theories demonstrated the effectiveness of the approach through an increase in reasoning abilities. Employing reuse with a translation from FOL to OWL created the opportunity to reuse a rigorous axiomatization of events that was able to provide useful semantics for the existing Semantic Web ontologies.

8.2 An Extended Definition of Reuse

The definition of reuse presented in Chapter 5 can be extended to account for the types of scenarios discussed here by allowing for signature and logical language translations.

Upon closer examination, we find that these approaches to reuse are in fact nearly accounted for by the definition of reusability as presented in Chapter 5 as follows:

Recall, Definition 12 states that $T$ is reusable for $M$ intended iff:

there is a mapping $\pi$ from some reduct of $M$ intended to a reduct of $Mod(T')$.

$$\pi : \text{Red}(M \text{ intended}, \sigma_1) \to \text{Red}(Mod(T'), \sigma_2)$$

where:

- $T' \leq T$ (i.e. $T$ is a nonconservative extension of $T'$, and $T$ and $T'$ are in the same hierarchy, as defined in the previous chapter.)
- $\sigma_1 \subseteq \sigma(M \text{ intended})$
- $\sigma_2 \subseteq \sigma(T')$

The mapping $\pi$ from the structures $\text{Red}(M \text{ intended}, \sigma_1)$ to the structures $\text{Red}(Mod(T'), \sigma_2)$ can also account for any required signature translations. What remains is to account for the relationship between the models if $T$ and $M \text{ intended}$ are in different logical languages. This requires that we consider a translation of the given theory into the required logical language. The resulting, extended definition of reusability is as follows:

Definition 39. Given some $T$ in logic $L_1$ and $M \text{ intended}$ in logic $L_2$.

$T$ is reusable-with-translations for $M \text{ intended}$ iff:

there exist theories $T', T''$ in logic $L_2$ such that:

- $T'$ is a logic translation of $T$ from $L_1$ to $L_2$, and

- there is a mapping $\pi$ from some reduct of $M \text{ intended}$ to a reduct of $Mod(T'')$.

$$\pi : \text{Red}(M \text{ intended}, \sigma_1) \to \text{Red}(Mod(T''), \sigma_2)$$

where:

- $T'' \leq T'$ (i.e. $T'$ is a nonconservative extension of $T''$, and $T'$ and $T''$ are in the same hierarchy, as defined in the previous chapter.)
In other words, in an extended definition of reuse, \( T \) is reusable-with-translations for \( M \) intended if there is a logic translation of \( T \) to the logical language of \( M \) intended that is reusable (as originally defined) for \( M \) intended. This definition extends similarly for the reusability of a set of theories: a set of theories is reusable-with-translations if each theory in the set is reusable-with-translations.

Similarly, we can extend the results from Theorem 4 regarding our intuitions of reusability with respect to the theories’ axiomatizations.

**Theorem 11.** If set of ontologies \( T_1, ..., T_n \) with logical languages \( L_1, ..., L_n \) and signatures \( \sigma(T_1), ..., \sigma(T_n) \) (respectively) is reusable-with-translations for \( M \) intended with logical language \( L_R \) and signature \( \sigma(M \) intended), then \( Th(M \) intended) contains nontrivial subtheories \( T_1'', ..., T_n'' \); where

- each \( T_1'', ..., T_n'' \) is a subtheory of \( T_1', ..., T_n' \), and
- each \( T_1', ..., T_n' \) is a logic translation of \( T_1'', ..., T_n'' \) from \( L_1 \rightarrow L_R, ..., L_n \rightarrow L_R \), and
- each \( T_1'', ..., T_n'' \) is a signature translation of \( T_1', ..., T_n' \) to \( \sigma(M \) intended).

**Proof.** If \( T_1, ..., T_n \) are reusable-with-translations for \( M \) intended, where \( M \) intended is in \( L_R \) and \( T_1, ..., T_n \) are in \( L_1, ..., L_n \), respectively, then by definition for each \( T_i \):

There exists a \( T_i' \) s.t. \( T_i' \) is in language \( L_R \) and \( T_i' \) under the logic mapping of \( L_R \) to \( L_i \) is a subtheory of \( T_i \), and \( T_i' \) is reusable for \( M \) intended.

Therefore, by Theorem 4, \( Th(M \) intended) contains nontrivial subtheories of each \( T_i' \).

We speculate that, similar to Theorem 4, this result is in fact provable in both directions, however it is not necessary for our purposes.

This proof indicates that the extended definition of reusability maintains our intuitions of reuse, while adding the conditions required to account for signature and logical language translations. The reuse operations must also be extended to account for these translations in order to capture the necessary operations with respect to the definition of a theory that is reusable-with-translations. Building on the classification presented in Chapter 5, we extend the operations by which an ontology may be reused; for reuse-with-translations, there are 6 distinct reuse operations by which an ontology may be manipulated:

The original 4 operations remain unchanged:

- As-is: \( as\_is(T) = T \)
- Extraction: \( extraction(T, T^-) = T/T^- \)
  Where the \( / \) symbol denotes the difference between two theories.
- Extension: \( extension(T, T^+) = T \cup T^+ \) where \( T^+ \notin S \)
  Where \( S \) is some ontology repository.
- Combination: \( combination(T_1, T_2)T_1 \cup T_2 \) where \( T_2 \in S \)
  Where \( S \) is some ontology repository.

And there are two new operations:
**Signature Translation of \( T \):** refers to the reuse of \( T \) via the application of mapping axioms to modify its signature.

**Definition 40.** signature_translation\((T, \pi_{\sigma(T_2)}) = \pi_{\sigma(T_2)}(T)\) where \( \pi_{\sigma(T_2)} \) is a signature translation into \( \sigma(T_2) \)
\( \pi : \sigma(T) \to \sigma(T_2) \)

**Logic Translation of \( T \):** refers reuse of \( T \) via a logic translation of \( T \) in language \( L_1 \) to some other logical language \( L_2 \).

**Definition 41.** logic_translation\((T, \alpha_{L_2}) = \alpha_{L_2}(T)\)
where \( \alpha_{L_2}(T) \) is a logic translation of \( T \) in \( L_1 \) to \( L_2 \).

As in Chapter 5, we might expect that the reuse-with-translation operations are commutative. While the result holds for the original 4 operations, since their definitions remain unchanged, this is not the case for the 2 new operations. The order in which a theory is translated to a different logic or signature, with respect to any other operations, will affect the order or signature of the operations to be applied. For example, consider the application of some signature-translation to \( T \), followed by some extraction.

This can be expressed as: \( \pi_{\sigma(T_2)}(T)/T^- \).

However, if we consider the application of some extraction operation to \( T \), followed by some signature-translation, in order to obtain the same result, the axioms removed \( T^- \) would need to be in a different signature. In other words, in the first case we would have \( \sigma(T^-) \subseteq \sigma(T_2) \), but in the second we would have \( \sigma(T^-) \subseteq \sigma(T) \). A similar situation arises when considering the order of application of the logic-translation operation.

We are now able to present an extension of Definition 19 for reuse-with-translations.

**Definition 42.** \( T_1, \ldots, T_n \) are reused-with-translations for \( M^{\text{intended}} \) iff

- \( T_1, \ldots, T_n \) are reusable-with-translations for \( M^{\text{intended}} \), and
- A signature-translation and/or logic-translation, followed by some set of reuse operations (as-is, extraction, extension, combination) applied to \( T_1, \ldots, T_n \) axiomatizes \( M^{\text{intended}} \).

Despite the lack of commutativity we can prove a result for the existence of reuse operations, similar to Theorem 11.

**Theorem 12.** Let \( S \) be some ontology repository, and let \( T_1, \ldots, T_n \) be some ontologies in \( S \). If \( T_1, \ldots, T_n \) are reusable-with-translations for \( M^{\text{intended}} \), then some set of reuse operations (as-is, extraction, extension, combination, signature translation, or logic translation) can be applied to axiomatize \( M^{\text{intended}} \).

**Proof.** Assume the theorem is false. Then we must have some \( T_1, \ldots, T_n \) reused-with-translations for \( R \) that cannot be described with some sequence of the reuse types.

There are 2 possible cases: reuse of a single theory or reuse of multiple theories.

**Case 1: reuse-with-translations of a single theory** \( n = 1 \) i.e. some \( T \) is reused-with-translations for \( \text{Th}(M^{\text{intended}}) \). Let \( \text{Th}(M^{\text{intended}}) \) have logical language \( L_R \) and let \( T \) have logical language \( L_T \).

By Theorem 11

There exists a theory \( T'' \) s.t. \( T'' \subseteq T' \) and \( T'' \subseteq \text{Th}(M^{\text{intended}}) \) such that: \( T' \) is the logic-translation of \( T \) from \( L_T \) to \( L_R \) of \( T \), i.e. \( \text{logic_translation}(T, \alpha_R) = T' \),

and \( T'' \) is the signature-translation from \( \sigma(T') \) to \( \sigma(\text{Th}(M^{\text{intended}})) \), i.e. \( \text{signature_translation}(T', \pi_{\text{Th}(M^{\text{intended}})}) = T'' \).
It is straightforward to see that after a logic-translation and a signature-translation are applied to $T$, one of the following situations must be true:

1. $T''' \subseteq T''$ and $T''' \subseteq Th(\mathcal{M}_{\text{intended}})$
2. $T''' \subseteq T''$ and $T''' = Th(\mathcal{M}_{\text{intended}})$
3. $T''' = T''$ and $T''' \subseteq Th(\mathcal{M}_{\text{intended}})$
4. $T''' = T''$ and $T''' = Th(\mathcal{M}_{\text{intended}})$

After accounting for the translations, the proof is straightforward as the possible cases reduce to that of Theorem 6. Therefore when $n = 1$ the assumption cannot hold so we can conclude that any reuse of a single theory can be defined by some sequence of reuse types.

**Case 2: reuse-with-translations of multiple theories** $n > 1$

Again, by Theorem 11
There exist theories $T_1'''$, ..., $T_n'''$ s.t. $T_1''' \subseteq T'''_1$ and $T_1''' \subseteq Th(\mathcal{M}_{\text{intended}})$, ..., $T_n''' \subseteq T'''_n$ and $T_n''' \subseteq Th(\mathcal{M}_{\text{intended}})$ where for each $T_i$ of $T_1$, ..., $T_n$.

There exists a theory $T_i'''$ s.t. $T_i''' \subseteq T'''_i$ and $T_i''' \subseteq Th(\mathcal{M}_{\text{intended}})$ such that: $T_i'$ is the logic-translation from $\mathcal{L}_{T_i}$ to $\mathcal{L}_R$ of $T_i$, i.e. logic_translation($T_i$, $\alpha_R$) = $T_i'$, and $T_i'''$ is the signature-translation from $\sigma(T_i')$ to $\sigma(Th(\mathcal{M}_{\text{intended}}))$, i.e. signature_translation($T_i'$, $\pi_{Th(\mathcal{M}_{\text{intended}})}$) = $T_i'''$

In order to reuse the translated theories, the combination operation must be applied at some point, to add each theory with the others (i.e. $n - 1$ times). We consider this necessary operation first, (recall that the operations are commutative) and then show how the result reduces to Case 1, for which we have shown the assumption cannot hold.

After a Signature Translation and a Logic Translation are applied to each $T_1$, ..., $T_n$ to obtain $T_1''$, ..., $T_n''$, consider all $n$ theories collectively, let $T'''_{new} = T_1'' \cup \ldots \cup T_n''$ i.e. $T_1'''$, ..., $T_n'''$ are reused for $T'''_{new}$ via combination.

Similarly, for all $T_1''' \subseteq T_1''$, ..., $T_n''' \subseteq T_n''$, $T'''_{new} = T_1'' \cup \ldots \cup T_n''$ and $T'''_{new} \subseteq Th(\mathcal{M}_{\text{intended}})$

By definition, $T'''_{new}$ is reused for $Th(\mathcal{M}_{\text{intended}})$ and we now have $n = 1$ for which we’ve shown the assumption is impossible (i.e. a subsequent sequence of operations exists for any scenario). Therefore we can conclude that any reuse of a set of $n$ theories where $n > 1$ can also be defined by some sequence of reuse types.

**8.3 Implications for the Solutions**

The previous section illustrated how the definitions and resulting theorems of reuse could be extended to account for reuse-with-translations. We now consider how the solutions presented earlier in this work support this broader notion of reuse.

**8.3.1 Signature Translation**

On the surface, the inclusion of signature translations for reuse indicates that our solutions will need to extend the methods they employ. For example, the notions of conservative and non-conservative extensions that were employed to assess the resulting benefits in Chapter [5] no longer have much relevance. If the candidate and the required theory are axiomatized in different signatures then neither one will be an extension of the other.
Similarly, the entailment-based approach to preference assessment, where a procedure evaluates the relationship between the required theory and two candidate theories loses its meaning. With disjoint signatures we cannot expect any entailment to be found, thus the results of many of the techniques presented previously appear to no longer be applicable.

A remedy for all of these challenges may be found in the solution for Search. In Chapter 3 we discussed the role of mappings from the requirements to a retrieved ontology as a conjecture of relevance. We also discussed the possible variation in search implementations, ranging from basic, trivial mappings to more exhaustive ones. In fact, the search procedure already provides a mechanism to account for reuse with signature translations. The candidate conjectures that are identified during search may be applied to attain a shared signature and, while any eventual reuse will in fact require a signature translation from the original theory, the subsequent solutions for reuse may be applied with the assumption of a shared signature.

8.3.2 Logical Language Translation

Since the solutions for the task of Search are oriented towards the signature matching and mapping, varying logical languages have no impact on the techniques. In fact, the conjectured mappings may be applied to the candidates to attain a common signature while the candidate remains in a different logical language from that of the required theory. The only notable implication would be an implementation complication as multiple sources and syntaxes would need to be accounted for within the task of Search.

The concept of reuse-with-translations between logical languages does have a considerable impact on the task of Choice. In particular, the procedures for assessing ontology preference presented in Chapter 4 are affected due to their dependence on the evaluation of reasoning problems. While the procedures presented are still feasible in a heterogeneous language scenario, it is a question of how to account for the potential variation in logics. One approach might be to simply assess the candidates in different languages separately. However, this precludes the consideration of merges between candidates axiomatized in different logics. It also adds to the complexity of the candidate assessment as it would result in multiple sets of preference orderings to consider. How is the designer to know how the theories compare between the logical languages? If a theory is the most preferred in OWL, there is no clear way for the developer to assess whether it is better or worse than the least preferred theory in FOL, and vice versa.

Simply comparing the results of reasoning problems across multiple logical languages is a challenge. The variation in a language’s expressivity has the potential to impact its ability to answer or even formalize a given query, thus there are additional dimensions to consider when evaluating semantic requirements across multiple logical languages. If a candidate is unable to entail a requirement - it could be a result of ‘missing axioms’ required to strengthen the semantics, alternatively it may be the case that it is not possible to add the necessary semantics in the logical language, or it could be that the requirement itself is not expressible.

These challenges would be simplified if all candidates were translated to a common language, however this only provides a single perspective on their potential reuse. Given that there are certain advantages and disadvantages to using different logical languages, if the logical language is not a strict development requirement, it is be advisable that the evaluation consider multiple logical languages in order to gain broader, more accurate assessment. Yet precisely how this should best be accomplished is unclear.

These issues are somewhat irrelevant assuming that the required logical language is known and static. In this case, the necessary approach would seem to be to simply translate all of the candidates to the same (required) logical language. While the results may differ if the theories and requirements are assessed in alternate logical languages, this is irrelevant if it is known that these languages are not viable options for development.
Regardless of the approach taken for CQ assessment, the solution will need to address the issue that the mappings between logical languages are not always straightforward. In some cases, such as translating from OWL to FOL, it may be possible to preserve all of the semantics of the original candidate. In other cases, semantics may be lost, and there also may be multiple possible translations. It is not always clear how to determine whether a particular translation from one language to another is ‘maximal’, or whether such a proof is even possible. A concern resulting from this is that when these translations are performed, they may result in somewhat misleading representations of a theory. If we translate a very expressive theory into a language with limited expressive abilities, there will be some loss of semantics. Unfortunately, it may not always be possible to tell whether semantics are necessarily lost due to properties of the logical languages, or whether they are lost due to the translations themselves. The lack of a definite means of translation introduces an undesirable element of variation and unpredictability for reuse.

Should the definition of reuse be extended to include logic translations, some strategy for consistently implementing translations between logics will be required for an effective assessment. The issue of how to validate such mappings remains an open problem.

8.4 Summary

While the concept of reuse with translations may not be part of the current paradigm of reuse, it is without question an important future consideration. These practices have the potential to be tremendously beneficial for the development of ontologies, both in expanding the possible candidate ontologies, and in allowing developers to take advantages of the strengths and weaknesses of different available logical languages. For example, development of a FOL ontology might take advantage of the widespread application and use of OWL ontologies, whereas as in our earlier example OWL ontologies may take advantage of the rigorous definitions that are more typical of FOL ontologies.

We have shown that reuse with signature translations may in fact be supported in the current state, via the solutions presented in the thesis. Reuse with logical language translations remains an open challenge that requires more advances to be made in the community. While the solutions provided in this thesis are capable of supporting this sort of reuse – it is the larger issue of mappings between logical languages must be resolved by the community to support an effective implementation. In the long term, when more progress has been made on the theory and application of logical language mappings not only will the issues of candidate assessment be resolved, but ideally the required theory may be comprised of multiple languages. Such an implementation would be supported by the solutions presented in this thesis, and in this case there would be no need to translate and evaluate candidates in a common logical language. We discuss this and other important directions for future work in the final chapter of the thesis.
Chapter 9

Conclusion

The vision for ontology reuse is to achieve a state in which reuse is not simply a norm of ontology design, but in which design via reuse occurs whenever it is possible. In this state the design of ontologies from-scratch does not occur unless no reusable ontology exists, thereby eliminating all redundant theories and unnecessary development efforts. More importantly, in such a state we can truly make a case for ontologies and their advantages over other technologies. Shareability will be possible between applications describing the same concepts, as originally envisioned. As the body of available ontologies grows and evolves, reuse will be possible in increasingly more applications, thereby continuously reducing the development work required for the application of ontologies.

This vision served as the motivation for the thesis. Towards this ideal state of reuse, we set out to achieve two key objectives that are necessary preconditions in order for the vision of reuse to be both feasible and encouraged: (1) to make design via reuse easier to perform, and (2) to ensure that the benefits of reuse are predictable for the developer. Solutions to both of these objectives are presented in the previous chapters of this thesis. In Chapters 3 and 4 we proposed a division of labour and a set of techniques to simplify design via reuse. Then in Chapter 5 we gave a formal definition of reuse and leveraged this to provide a means for ensuring that the resulting benefits of reuse are transparent and tractable. We defined the necessary requirements for an infrastructure to support the implementation of these solutions; these also serve as an architecture that prescribes how the contributions work together in the process of design via reuse. By achieving the objectives, we have contributed a formal definition of reuse, rigorous techniques to support the various tasks of design via reuse, and the design of a comprehensive architecture for the solutions’ implementation. We summarize each of these contributions in the following section.

9.1 Contributions

To truly define reuse required the consideration of a combination of logical and extralogical conditions. From a logical perspective, a theory must be reusable to be reused – to express this, we have provided a formal definition of reusability. But reuse is not simply a property of a theory; if a theory is reused then some operations must have been performed on it – to capture this, we have provided a characterization of operations for reuse. The resulting definition of reuse is constructed with a combination of the conditions of reusability and the application of some reuse operations. Not only is the definition novel, it is the first formal definition for reuse that has ever been presented. The lack of such a definition has been inhibiting development in the domain of ontology reuse. This contribution provides clear context in which past and future contributions may be situated. More importantly, the formal definition of the concepts of reuse, reusability, and the precise identification of reuse
operations are necessary tools to bring the process of reuse, and ontology development in general, closer to an engineering discipline. We have also generalized the definitions presented here to include both signature and logical language translations. The consideration of reuse-with-translations is novel in its own right. Further, it is a crucial consideration in order to ensure continued support for reuse as advances are made in the ontology community. Owing to their rigorous foundation, the solutions provided in this thesis remain applicable and are easily extended for this generalized definition of reuse.

As a means of designing rigorous techniques for reuse, we have specified a formal definition of a natural intuition of preference between ontologies. No attempt has ever been made to formalize this concept, thus we find that this definition addresses a significant, open problem in the ontology community. While we may have had some intuition of preference prior to this, intuition alone is insufficient to address the difficulty of choosing between candidates. The formal definition provides a well-founded, precise way of comparing candidate ontologies with respect to the requirements. In addition to the formal definition, we have provided a set of procedures capable of assessing preference. These were crucial in order to ensure that the concept of preference would be a viable solution in practice. Not only did we demonstrate the viability of the preference ordering, we have illustrated that the procedures are effective in practice by demonstrating the techniques on real ontologies. The definition of reuse also served as a source of techniques. We have shown how the definition of reusability could be extended to assess whether a given ontology is reusable with respect to some requirements. This sort of analysis has not been attempted before, and provides valuable, detailed insight for the developer. We have described how the definitions of the reuse operations could be leveraged to determine precisely what operations are required to reuse a given ontology. While various algebras have been proposed for ontologies, none have been employed in this way. The analysis leverages the operations both as a guide for the reuse of an ontology as well as a tool for the developer to reliably assess the outcomes of reuse.

While the notion of an architecture, whether for reuse or any other more general approach to ontology development, is not particularly original – the architecture that we present is novel in terms of the solutions it incorporates. It is also of critical importance for the work presented here. We have provided a comprehensive description of how the solutions provided here work in concert to support the process of reuse. Further, we have outlined the requirements that must be met by any infrastructure in order to support the solutions. This provides a critical foundation from which future work may continue to pursue the vision of reuse.

9.2 Future Work

While in theory, the solutions presented here support the extended definition of reuse with logical language translations, developments in the techniques and tools available to perform these translations are necessary before this becomes a feasible option for reuse. In the current state, we must compromise by translating all candidates to a particular logical language for a uniform assessment, however efforts such as [46, 58] are underway to address the challenge of reasoning across logical languages. In a future state when such tools are more mature, there will be no need to translate candidate ontologies in order to assess them. It is also interesting to note that in this case we might conceptualize the required theory itself as being comprised of multiple logical languages, and that this scenario is also easily covered with a minor extension to the definitions presented in Chapter 8.

Other directions for future work relate primarily to the implementation of the solution. This should be relatively straightforward given the level of detail provided, however the effort required will certainly vary depending on the context of the implementation: for example the logical language(s) included, the community, and the general purpose tools available for its development. In addition, it is important to note that we have presented the
theoretical solutions; some implementations may require the use of heuristics or approximations (for example to implement the procedures) in order for the solution to be practical. Further, in order to augment rather than detract from this contribution it is of major importance that the usability of the implementation be considered. A well-designed solution that is not usable for developers will not serve to address the objectives at all.

Finally, it will be interesting for future work to consider other applications of the solutions presented here. In Chapter 2 we discussed the parallels often drawn between ontology and software design. While we found existing work in the software community was unable to address the challenges of ontology reuse, it would be an interesting exercise to consider whether any analogies may be drawn from these definitions and techniques to be applied in the software community. Alternatively, perhaps ontology representations of software functionality and requirements could be employed with this particular methodology to aid in the selection of appropriate software. Another closely related area that could benefit in the application of these solutions is the problem of shareability. Whereas ontology reuse considers the problem of designing an ontology from some existing theories, shareability considers the problem of determining whether or how much an (existing) ontology can share with some other ontologies. While reuse is related in that it is meant to support shareability, the two problems are distinct; unless it is known that two ontologies have reused the same theory, it is non-trivial to assess the shareability between them. We speculate that an approach similar to our assessment of ontology preference might be applied to assess the relative shareability of a set of ontologies. This too would be an interesting direction for future work.

9.3 Final Thoughts

The thesis presented here has filled a major void in the field of ontology development. Where no clear understanding of reuse was demonstrated before, we have provided a formal definition. Where means of design via reuse were ad-hoc and aided only by guidelines or imprecise metrics, we have provided a set of rigorous techniques to support the tasks involved and defined the infrastructure required to support the process. Through this, we have provided the ontology community with the tools to achieve the vision of reuse. Most importantly, we have moved the practice of ontology reuse, and ontology development in general, closer to an engineering discipline.
Bibliography


Appendix A

Supplementary Procedures

A.1 Modularization

Excerpt from [36]:

In this section, we outline a semi-automated procedure for using the relationships among core theories and complex theories within the repository to decompose an ontology into irreducible modules.

**Procedure 9** *Decomp*(R, T, M)

**Require:** Core repository R = ⟨R, ⊑⟩, theory T.

**Ensure:** M is the modularization of T

- M ← ∅
- T ← ∅
- FindTheory(R, T, T)
  - Π ← translation definitions for theories in T into T
  5: if T ∪ Π |= T and T ∪ Π is a conservative extension of T then
    - for all S_i ∈ T do
      - T_i ← ∅
      - for all σ ∈ S_i do
        - σ^r ← translation of σ
      - T_i ← T_i ∪ {σ^r}
    - M ← M ∪ {T_i}
  10: end for
  15: if T ∪ Π ∪ T consistent then
    - T is reducible to an extension of theories in T
  else
    - T is not reducible to any set of theories in R
  end if
  20: end if

A.2 Addition to Hierarchy

Excerpt from [36]
Procedure 10 FindTheory($\mathbb{R}, T, T$)

Require: Core repository $\mathbb{R} = (\mathbb{R}, \sqsubseteq)$, theory $T$.

Ensure: $T_i \in T$ is a maximal theory in $C_i$ that is interpreted by $T$, for each core hierarchy $C_i \in \mathbb{R}$.

$T \leftarrow \emptyset$

for all $C_i \in \mathbb{R}$ whose root theories are interpretable by $T$ do

$\Delta_i \leftarrow$ translation definitions for $T$ into theories in $C_i$

$Chains_i \leftarrow \text{ChainDecomp}(C_i)$

for all $G_{ij} \in Chains_i$ do

Candidate$_{ij} \leftarrow \emptyset$

$T_{\text{current}} \leftarrow$ minimal theory in $G_{ij}$

$T_{\text{max}} \leftarrow$ maximal theory in $G_{ij}$

while $T_{\text{current}} \neq T_{\text{max}}$ do

if $T \cup \Delta_i \models T_{\text{current}}$ and $T \cup \Delta_i$ is a conservative extension of $T_{\text{current}}$ then

Candidate$_{ij} \leftarrow T_{\text{current}}$

$T_{\text{current}} \leftarrow \text{NextTheory}(G_{ij}, T_{\text{current}})$

else

$T_{\text{current}} \leftarrow T_{\text{max}}$

end if

end while

end for

Collect$_i \leftarrow \bigcup_j \{\text{Candidate}_{ij}\}$

PosetSort(Collect$_i$, $P$)

$T_i \leftarrow$ set of axioms in the union of maximal theories in $P$

$T \leftarrow T \cup \{T_i\}$

end for

Here we present a semi-automated procedure that inserts a new theory into a closed, atomistic repository so that the resulting repository is still closed and atomistic. This procedure is intended to be used with the core hierarchies which are required to be closed and atomistic in our repository, but is not restricted to those. In general, this leads to the introduction of new trunk theories into the (core) hierarchy, as well as the refinement of some former trunk theories in the original (core) hierarchy.

Adding a theory $T$ to an existing closed, atomistic hierarchy presumes, of course, that the signature of $T$ is the same as the signature of the theories in the hierarchy, and that $T$ is an extension of the root theory of the hierarchy. Procedures 11, 12, and 13 give the full details. The following notation is used throughout the procedures:

$\text{root}(\mathbb{C})$: Reference to the root theory of the closed, atomistic hierarchy $\mathbb{C}$.

$\text{trunk}(\mathbb{C})$: Reference to the set of trunk theories of the closed, atomistic hierarchy $\mathbb{C}$.

$\text{profile}(T)$: Reference to the set of theories that constitute the profile of the theory $T$.

$\text{similarity}(T_1, T_2)$: Returns a theory $T$ that is the similarity of the theories $T_1$ and $T_2$ in the sense of Definition 43.

$\text{difference}(T_1, T_2)$: Returns a theory $T$ that is the difference between the theories $T_1$ and $T_2$ in the sense of Definitions 44 and 45.

Definition 43. Let $T_1$ and $T_2$ be theories in the same hierarchy with the signature $\Sigma$. 
The similarity between $T_1$ and $T_2$ is the strongest theory (up to logical equivalence) $S \subseteq T_1 \cap T_2$ with $\Sigma(S) = \Sigma(T_1)$ so that for all $\sigma, \omega \in L(T_1)$ if

$$T_1 \models \sigma \text{ and } T_2 \models \omega \text{ and } S \models \sigma \text{ and } S \models \omega$$

then either $\sigma \lor \omega$ is independent of $S$ or $\sigma \lor \omega$ is a tautology.

**Definition 44.** Let $T_1, T_2$ be theories in the same hierarchy such that $T_1 < T_2$.

The remainder between $T_2$ and $T_1$ is the weakest theory (up to logical equivalence) $T'$ with $\Sigma(T') = \Sigma(T_1)$ so that

$$T_2 \equiv T_1 \cup T'.$$

**Definition 45.** Let $T_1, T_2$ be theories in the same closed hierarchy $\mathbb{H}$ so that $T_1 < T_2$. Let $T'$ be the remainder between $T_2$ and $T_1$ and let $T'_{\text{root}}$ be the unique root theory of $\mathbb{H}$.

The difference between $T_2$ and $T_1$ is the theory (up to logical equivalence) $T_d$ with $\Sigma(T_d) = \Sigma(T_1)$ so that

$$T_d \equiv T'_{\text{root}} \cup T'.$$

**Procedure 11 UpdateHierarchy(C, T)**

Require: A closed, atomistic hierarchy $C = \langle C, \leq \rangle$, and a theory $T$ such that $\Sigma(C) = \Sigma(T)$ and $T \models \text{root}(C)$.

Ensure: $C = \langle C \cup T, \leq \rangle$ is a closed, atomistic hierarchy.

* UpdateProfile(C, T)
* $C \leftarrow C \cup T$
* CloseHierarchy(C, T)

**Procedure 12 UpdateProfile(C, T)**

Require: A closed, atomistic hierarchy $C = \langle C, \leq \rangle$, a theory $T$ such that $\Sigma(C) = \Sigma(T)$ and $T \models \text{root}(C)$.

Ensure: $T$ has a profile in $C$, $C$ is atomistic and the similarity between any two trunk theories is a root theory.

* Covered $\leftarrow \emptyset$
* $T^d \leftarrow T$ (the difference between $T$ and the currently covered trunk theories)
* while $\exists T' \in \text{trunk}(C) \setminus \text{Covered}$ with similarity$(T', T^d) \neq \text{root}(C)$ do
  * $S \leftarrow \text{similarity}(T', T^d)$
  * Covered $\leftarrow \text{Covered} \cup S$
  * $T^d \leftarrow \text{difference}(T^d, S)$
  * $T'^d \leftarrow \text{difference}(T', S)$
  * $\text{trunk}(C) \leftarrow (\text{trunk}(C) \setminus T') \cup \{S, T'^d\}$
  * profile$(T') \leftarrow \{S, T'^d\}$
* for all $T_i \in C$ do
  * if $T' \in \text{profile}(T_i)$ then
    * profile$(T_i) \leftarrow (\text{profile}(T_i) \setminus T') \cup \text{profile}(T')$
  * end if
* end for
* trunk(C) $\leftarrow \text{trunk}(C) \cup T^d$
* profile$(T) \leftarrow \text{Covered} \cup T^d$
Procedure 13 CloseHierarchy($C, T$)

**Require:** An atomistic hierarchy $C = (C, \leq)$ and a theory $T$ such that $T \in C$, $T$ has a profile in $C$ and $C' = (C \setminus T, \leq)$ is closed.

**Ensure:** $C = (C \cup T, \leq)$ is a closed, atomistic hierarchy.

```plaintext
for all $T_i \in C$ do
    profile($S$) ← profile($T$) \cap profile($T_i$)  \hspace{1cm} \text{(Similarity)}
    S ← \bigcup_{j} S_j \in profile($S$)
    if $S \notin C$ then
        $C ← C \cup S$
        CloseHierarchy($C, S$)
    end if
end for
```
Appendix B

Comparing Participation Ontologies: Proof Files

Note: all files are specified in Prover9 syntax.

B.1 Theories Modified for Common Hierarchy

\( H_{TR} \)

% katsumi-thesis-searchcase-semantic-regts-v1.clif
%minor correction to CQ4 (no impact on results, only changes signature)
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
   (exists z
    tr_has_role(x,z,t))))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
   tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
  (tr_participates_in(x,a_1,t)
& tr_part_of(a_1,a_2)
->
    tr_participates_in(x,a_2,t)).

%CQ4 'Objects can not participate in more than one different activity.'
(all x all a_1 all a_2
  (tr_object(x)
    & tr_participates_in(x,a_1,t)
    & tr_participates_in(x,a_2,t)
    ->
      ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
  (tr_participates_in(x,a,t)
    ->
      (tr_active_in(x,a,t) | tr_passive_in(x,a,t) | tr_consumed_in(x,a,t)))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) 
                  ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t))))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t))))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2)))).
(all x (tr_person(x) -> tr_person(x)))).
(all x (tr_object(x) -> tr_object(x)))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t)))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t)))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t)))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x)))).
(all x (g_Object(x) -> g_Object(x)))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y)))).
(all x all y (g_pre(x,y) -> g_pre(x,y)))).
(all y (g_Time(y) -> g_Time(y)))).
%psl_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).
% Reading from file /stl/cchui/macroleod-master/qs/test/conversions/
  dolece_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

H_PSL_participates

%psl_participates for common hierarchy
%psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).
(all x (psl_object(x) -> ~psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) &
   psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) &
  psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) &
  psl_is_occurring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) &
  psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(
  t2) & (psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) &
  psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(
  x),t,psl_endof(x)))).
(all occ all t (psl_is_occurring_at(occ,t) <-> psl_activity_occurrence(occ)
  & psl_betweenEq(beginof(occ),t,endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(
  x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(
  o_1,o_2))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
%katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)
))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2)))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).
\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) \rightarrow Q(x))).
(all t t1 t2 (SUM(t,t1,t2) \rightarrow SUM(t,t1,t2))).

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z y (O(z,y) \rightarrow O(z,y))).
(all z y (DJ(z,y) \rightarrow DJ(z,y))).
(all x y (PP(x,y) \rightarrow PP(x,y))).
(all x y (U(x,y) \rightarrow U(x,y))).
(all x (AtP(x) \rightarrow AtP(x))).
(all x y (U(x,y) \rightarrow U(x,y))).

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) \rightarrow AB(x))).
(all x (PED(x) \rightarrow PED(x))).
(all x (NPED(x) \rightarrow NPED(x))).
(all x (AS(x) \rightarrow AS(x))).
(all x (EV(x) \rightarrow EV(x))).
(all x (STV(x) \rightarrow STV(x))).
(all x (TQ(x) \rightarrow TQ(x))).
(all x (PQ(x) \rightarrow PQ(x))).
(all x (AQ(x) \rightarrow AQ(x))).
(all x (R(x) \rightarrow R(x))).
(all x (M(x) \rightarrow M(x))).
(all x (F(x) \rightarrow F(x))).
(all x (POB(x) \rightarrow POB(x))).
(all x (NPOB(x) \rightarrow NPOB(x))).
(all x (ACH(x) \rightarrow ACH(x))).
(all x (ACC(x) \rightarrow ACC(x))).
(all x (ST(x) \rightarrow ST(x))).
(all x (PRO(x) \rightarrow PRO(x))).
(all x (TL(x) \rightarrow TL(x))).
(all x (SL(x) \rightarrow SL(x))).
(all x (TR(x) \rightarrow TR(x))).
(all x (PR(x) \rightarrow PR(x))).
(all x (AR(x) \rightarrow AR(x))).
(all x (APO(x) \rightarrow APO(x))).
(all x (NAPO(x) \rightarrow NAPO(x))).
(all x (MOB(x) \rightarrow MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

\textbf{B.1.1} \textit{H-DOLCE participation}

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t)))).
(all x (ED(x) -> (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t))).
(all x all y all t (PC(x,y,t) <-> (all t1 (P(t1,t) -> PC(x,y,t1))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/   dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t)))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t,t1,t2) -> PRE(x,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/   dolce_time_mereology.p9
(all x all y (P(x,y) -> T(y) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y)))).
(all x all y (T(x) -> P(x,x))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z)))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (T(z) & P(z,x) & -O(z,y)))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z))))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z))))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z)))))).

(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).

(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y)))))))).

(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y))))))))).

(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/ dolce_taxonomy.p9

(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x)))

(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).

(all x (EV(x) | STV(x) -> PD(x))).

(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).

(all x (R(x) -> AB(x))).

(all x (M(x) | F(x) | POB(x) -> PED(x))).

(all x (NPOB(x) -> NPED(x))).

(all x (ACH(x) | ACC(x) -> EV(x))).

(all x (ST(x) | PRO(x) -> STV(x))).

(all x (TL(x) -> TQ(x))).

(all x (SL(x) -> PQ(x))).

(all x (TR(x) | PR(x) | AR(x) -> R(x))).

(all x (APO(x) | NAPO(x) -> POB(x))).

(all x (MOB(x) | SOB(x) -> NPOB(x))).

(all x (I(x) -> TR(x))).

(all x (S(x) -> PR(x))).

(all x (ASO(x) | NASO(x) -> SOB(x))).

(all x (SAG(x) | SC(x) -> ASO(x))).

(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).

(all x (ED(x) -> -PD(x) & -Q(x) & -AB(x))).

(all x (PD(x) -> -Q(x) & -AB(x))).

(all x (Q(x) -> -AB(x))).

(all x (ED(x) <-> PED(x) | NPED(x) | AS(x))).

(all x (PED(x) -> -NPED(x) & -AS(x))).

(all x (NPED(x) -> -AS(x))).

(all x (PD(x) <-> EV(x) | STV(x))).

(all x (EV(x) -> -STV(x))).

(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x))).

(all x (TQ(x) -> -PQ(x) & -AQ(x))).

(all x (PQ(x) -> -AQ(x))).
(all x (PED(x) <-> M(x) | F(x) | POB(x))).
(all x (M(x) -> -F(x) & -POB(x))).
(all x (F(x) -> -POB(x))).
(all x (EV(x) <-> ACH(x) | ACC(x))).
(all x (ACH(x) -> -ACC(x))).
(all x (STV(x) <-> ST(x) | PRO(x))).
(all x (ST(x) -> -PRO(x))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> -PR(x) & -AR(x))).
(all x (PR(x) -> -AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> -NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> -SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> -NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> -SC(x))).

%*****relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

%'no dolce_participation equivalent for tr_part_of'

%'no dolce_participation equivalent for tr_object but we are given the
additional info that an object is an endurant’)
%(all a (tr_object(a) -> ED(a))).

%'no dolce_participation equivalent for tr_active_in'

%'no dolce_participation equivalent for tr_passive_in’
"no dolce_participation equivalent for tr_consumed_in"

*********root theory
%Root Theory (collection of candidates' signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)) .
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

B.1.2 H_gangemi_participation

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/gangemi.p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (exists y g_pre(x,y))))).
(all x (g_pre(x,y) -> (g_Object(x) | g_Event(x) | g_Time(y)))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory:%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)
  ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t))))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% _g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%_psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,
  occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))
  .
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
\[(\forall y \ (PD(y) \rightarrow PD(y)))\].
\[(\forall t \ (T(t) \rightarrow T(t)))\].
\[(\forall x \ \forall t \ (PRE(x,t) \rightarrow PRE(x,t)))\].
\[(\forall t1 \ \forall t \ (P(t1,t) \rightarrow P(t1,t)))\].

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
doche_present.p9
\[(\forall x \ (Q(x) \rightarrow Q(x)))\].
\[(\forall t \ \forall t1 \ \forall t2 \ (SUM(t,t1,t2) \rightarrow SUM(t,t1,t2)))\].

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
doche_time_mereology.p9
\[(\forall z \ \forall y \ (O(z,y) \rightarrow O(z,y)))\].
\[(\forall z \ \forall y \ (DJ(z,y) \rightarrow DJ(z,y)))\].
\[(\forall x \ \forall y \ (PP(x,y) \rightarrow PP(x,y)))\].
\[(\forall x \ \forall y \ (U(x,y) \rightarrow U(x,y)))\].
\[(\forall x \ (AtP(x) \rightarrow AtP(x)))\].
\[(\forall x \ \forall y \ (U(x,y) \rightarrow U(x,y)))\].

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
doche_taxonomy.p9
\[(\forall x \ (AB(x) \rightarrow AB(x)))\].
\[(\forall x \ (PED(x) \rightarrow PED(x)))\].
\[(\forall x \ (NPED(x) \rightarrow NPED(x)))\].
\[(\forall x \ (AS(x) \rightarrow AS(x)))\].
\[(\forall x \ (EV(x) \rightarrow EV(x)))\].
\[(\forall x \ (STV(x) \rightarrow STV(x)))\].
\[(\forall x \ (TQ(x) \rightarrow TQ(x)))\].
\[(\forall x \ (PQ(x) \rightarrow PQ(x)))\].
\[(\forall x \ (AQ(x) \rightarrow AQ(x)))\].
\[(\forall x \ (R(x) \rightarrow R(x)))\].
\[(\forall x \ (M(x) \rightarrow M(x)))\].
\[(\forall x \ (F(x) \rightarrow F(x)))\].
\[(\forall x \ (POB(x) \rightarrow POB(x)))\].
\[(\forall x \ (NPOB(x) \rightarrow NPOB(x)))\].
\[(\forall x \ (ACH(x) \rightarrow ACH(x)))\].
\[(\forall x \ (ACC(x) \rightarrow ACC(x)))\].
\[(\forall x \ (ST(x) \rightarrow ST(x)))\].
\[(\forall x \ (PRO(x) \rightarrow PRO(x)))\].
\[(\forall x \ (TL(x) \rightarrow TL(x)))\].
\[(\forall x \ (SL(x) \rightarrow SL(x)))\].
\[(\forall x \ (TR(x) \rightarrow TR(x)))\].
\[(\forall x \ (PR(x) \rightarrow PR(x)))\].
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

B.2 Proofs to Determine Hierarchy Ordering

To identify other possible orderings between the theories in the hierarchy, we assess whether $T_i \models T_j$ for any pair of theories. Because the mappings weren’t syntactically applied to each theory, we include the mapping axioms for any theories involved in the antecedent of a given entailment problem to attain the shared signature. Proof or model output included where applicable.

B.2.1 $H_{PSL}.participates \models H_{DOLCE}.participation$

Proof attempt for axiom 1 of DOLCE.participation:

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
    psl_participates.p9
%psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).
(all x (psl_object(x) -> -psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) & psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) & psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) & psl_is_occurring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) & psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) & (psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) & psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,psl_endof(x))).
(all occ all t (psl_is_occuring_at(occ,t) <-> psl_activity_occurrence(occ) & psl_betweenEq(beginof(occ),t,endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
%psl_prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

%dolce_2_Tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs

% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

%'no dolce_participation equivalent for tr_part_of'
%‘no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant’\%
(all a (tr_object(a) -> ED(a))).

%‘no dolce_participation equivalent for tr_active_in’

%‘no dolce_participation equivalent for tr_passive_in’

%‘no dolce_participation equivalent for tr_consumed_in’

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-repts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
% psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).
(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
-((all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).

Counterexample found:

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   psl_participates.p9
%psl_prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).
(all x (psl_object(x) -> -psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_activity_occurrence(o) &
   psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) &
    psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) &
    psl_is_occurring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) &
    psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) & (psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) &
    psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,psl_endof(x)))).
(all occ all t (psl_is_occurring_at(occ,t) <-> psl_activity_occurrence(occ) &
    psl_betweenEq(beginof(occ),t,psl_endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
%psl_prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

%dolce_2_Tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs

% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

% 'PD equivalent to tr_event'
(all a  (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t  (PRE(a,t) <-> tr_exists_at(a,t))).

%'no dolce_participation equivalent for tr_part_of'

%'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant')
(all a  (tr_object(a) -> ED(a))).

%'no dolce_participation equivalent for tr_active_in'

%'no dolce_participation equivalent for tr_passive_in'

%'no dolce_participation equivalent for tr_consumed_in'

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a  (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x  (g_Event(x) -> g_Event(x))).
(all x  (g_Object(x) -> g_Object(x))).
(all x all y  (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y  (g_pre(x,y) -> g_pre(x,y))).
(all y  (g_Time(y) -> g_Time(y))).
%psl_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
-((all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).
H_{PSL\_participates} \models H_{gangemi\_participation}

**Proof attempt for axiom 1 of gangemi\_participation:**

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  psl_participates.p9
% psl_ prefix added to all terms to distinguish from other candidates

(all x (psl\_activity\_occurrence(x) -> -psl\_object(x) & -psl\_timepoint(x))).
(all x (psl\_object(x) -> -psl\_timepoint(x))).
(all o all t (psl\_is\_occurring\_at(o,t) -> psl\_activity\_occurrence(o) & psl\_timepoint(t))).
(all x all t (psl\_exists\_at(x,t) -> psl\_object(x) & psl\_activity\_occurrence(x) & psl\_timepoint(t))).
(all x all occ all t (psl\_participates\_in(x,occ,t) -> psl\_object(x) & psl\_exists\_at(x,t) & psl\_is\_occurring\_at(occ,t))).
(all t1 all t2 all t3 (psl\_between(t1,t2,t3) <-> psl\_before(t1,t2) & psl\_before(t2,t3))).
(all t1 all t2 (psl\_beforeEq(t1,t2) <-> psl\_timepoint(t1) & psl\_timepoint(t2) & (psl\_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl\_betweenEq(t1,t2,t3) <-> psl\_beforeEq(t1,t2) & psl\_beforeEq(t2,t3))).
(all x all t (psl\_exists\_at(x,t) <-> psl\_object(x) & psl\_betweenEq(beginof(x),t,psl\_endof(x))).
(all occ all t (psl\_is\_occurring\_at(occ,t) <-> psl\_activity\_occurrence(occ) & psl\_betweenEq(beginof(occ),t,psl\_endof(occ)))).

%relevance mappings to requirements
% psl\_2\_Tr
% psl\_ prefix added to all terms to distinguish from other candidates

% 'participates\_in equivalent to tr\_participates\_in'
(all x all a all t (psl\_participates\_in(x,a,t) <-> tr\_participates\_in(x,a,t))).

% 'activity\_occurrence equivalent to tr\_event'
(all a (psl\_activity\_occurrence(a) <-> tr\_event(a))).

% 'no psl equivalent for tr\_has\_role'

% 'exists\_at equivalent to tr\_exists\_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of (o_1,o_2))).

%gangemi_2_Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature

(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t))))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))
.
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (FQ(x) -> FQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
-((all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y)))))))
.

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2
A
PPEndix

B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[ t = 0. \]
\[ c1 = 0. \]
\[ \text{beginof}(0) = 0. \]
\[ \text{beginof}(1) = 0. \]
\[ \text{endof}(0) = 0. \]
\[ \text{endof}(1) = 0. \]
\[ \text{psl\_beginof}(0) = 0. \]
\[ \text{psl\_beginof}(1) = 0. \]
\[ \text{psl\_endof}(0) = 0. \]
\[ \text{psl\_endof}(1) = 0. \]
\[ f1(0,0) = 0. \]
\[ f1(0,1) = 0. \]
\[ f1(1,0) = 0. \]
\[ f1(1,1) = 0. \]
\[ g\_Event(0). \]
\[ - g\_Event(1). \]
\[ - g\_Object(0). \]
\[ - g\_Object(1). \]
\[ \text{psl\_activity\_occurrence}(0). \]
\[ - \text{psl\_activity\_occurrence}(1). \]
\[ - \text{psl\_object}(0). \]
\[ - \text{psl\_object}(1). \]
\[ - \text{psl\_timepoint}(0). \]
\[ - \text{psl\_timepoint}(1). \]
\[ \text{tr\_event}(0). \]
\[ - \text{tr\_event}(1). \]
\[ - \text{tr\_object}(0). \]
\[ - \text{tr\_object}(1). \]
- g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).

- g_pre(0,0).
- g_pre(0,1).
- g_pre(1,0).
- g_pre(1,1).

- psl_before(0,0).
- psl_before(0,1).
- psl_before(1,0).
- psl_before(1,1).

- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(1,0).
- psl_exists_at(1,1).

- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).

- psl_between(0,0,0).
- psl_between(0,0,1).
- psl_between(0,1,0).
- psl_between(0,1,1).
- psl_between(1,0,0).
- psl_between(1,0,1).
- psl_between(1,1,0).
- psl_between(1,1,1).

- psl_betweenEq(0,0,0).
- psl_betweenEq(0,0,1).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).

\[H_{\text{PSL.participates}} \models H_{T_R}\]

**Proof attempt for CQ1 of \(T_R\):**

%common hierarchy
(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).
(all x (psl_object(x) -> -psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) & psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) & psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) & psl_is_occurring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) & psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) & (psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) & psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,endof(x)))).
(all occ all t (psl_is_occurring_at(occ,t) <-> psl_activity_occurrence(occ) & psl_betweenEq(beginof(occ),t,endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).
% root theory
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).

(all a (tr_event(a) -> tr_event(a))).

(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).

(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).

(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).

(all x (tr_object(x) -> tr_object(x))).

(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).

(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).

(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

% gangemi.p9 signature

% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).

(all x (g_Object(x) -> g_Object(x))).

(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).

(all x all y (g_pre(x,y) -> g_pre(x,y))).

(all y (g_Time(y) -> g_Time(y))).

% psl_participation.p9 signature

% psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).

(all x (psl_object(x) -> psl_object(x))).

(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).

(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).

(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).

(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).

(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).

(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).

(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).

(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).

(all x psl_endof(x) = psl_endof(x)).

(all x psl_beginof(x) = psl_beginof(x)).

% dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ1 'Anything that participates in some event has some role when they are participating.'
-((all x all a all t
   (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
     (exists z
       tr_has_role(x,z,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

t = 0.

c1 = 0.

c2 = 1.

c3 = 2.

beginof(0) = 2.
BEGINOF(1) = 2.
BEGINOF(2) = 0.

ENDOF(0) = 0.
ENDOF(1) = 2.
ENDOF(2) = 0.

PSL_BEGINOF(0) = 0.
PSL_BEGINOF(1) = 0.
PSL_BEGINOF(2) = 0.

PSL_ENDOF(0) = 2.
PSL_ENDOF(1) = 0.
PSL_ENDOF(2) = 0.

- PSL_ACTIVITY_OCCURRENCE(0).
  PSL_ACTIVITY_OCCURRENCE(1).
  - PSL_ACTIVITY_OCCURRENCE(2).
  
  PSL_OBJECT(0).
  - PSL_OBJECT(1).
  - PSL_OBJECT(2).

- PSL_TIMEPOINT(0).
  - PSL_TIMEPOINT(1).
    PSL_TIMEPOINT(2).

- TR_EVENT(0).
  TR_EVENT(1).
  - TR_EVENT(2).

- PSL_BEFORE(0,0).
  - PSL_BEFORE(0,1).
  - PSL_BEFORE(0,2).
  - PSL_BEFORE(1,0).
  - PSL_BEFORE(1,1).
    PSL_BEFORE(1,2).
  - PSL_BEFORE(2,0).
    PSL_BEFORE(2,1).
    - PSL_BEFORE(2,2).

- PSL_BEFOREEQ(0,0).
  - PSL_BEFOREEQ(0,1).
- psl_beforeEq(0,2).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).
- psl_beforeEq(1,2).
- psl_beforeEq(2,0).
- psl_beforeEq(2,1).
- psl_beforeEq(2,2).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(0,2).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).

- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(0,2).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).
- psl_is_occuring_at(1,2).
- psl_is_occuring_at(2,0).
- psl_is_occuring_at(2,1).
- psl_is_occuring_at(2,2).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(2,0).
- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).

- psl_between(0,0,0).
- psl_between(0,0,1).
- psl_between(0,0,2).
- psl_between(0,1,0).
- psl_between(0,1,1).
- psl_between(0,1,2).
- psl_between(0,2,0).
- psl_between(0,2,1).
- psl_between(0,2,2).
- psl_between(1,0,0).
- psl_between(1,0,1).
- psl_between(1,0,2).
- psl_between(1,1,0).
- psl_between(1,1,1).
- psl_between(1,1,2).
- psl_between(1,2,0).
- psl_between(1,2,1).
- psl_between(1,2,2).
- psl_between(2,0,0).
- psl_between(2,0,1).
- psl_between(2,0,2).
- psl_between(2,1,0).
- psl_between(2,1,1).
- psl_between(2,1,2).
- psl_between(2,2,0).
- psl_between(2,2,1).
- psl_between(2,2,2).
- psl_betweenEq(0,0,0).
- psl_betweenEq(0,0,1).
- psl_betweenEq(0,0,2).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(0,1,2).
- psl_betweenEq(0,2,0).
- psl_betweenEq(0,2,1).
- psl_betweenEq(0,2,2).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,0,2).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).
- psl_betweenEq(1,1,2).
- psl_betweenEq(1,2,0).
- psl_betweenEq(1,2,1).
- psl_betweenEq(1,2,2).
- psl_betweenEq(2,0,0).
- psl_betweenEq(2,0,1).
- psl_betweenEq(2,0,2).
- psl_betweenEq(2,1,0).
- psl_betweenEq(2,1,1).
- psl_betweenEq(2,1,2).
- psl_betweenEq(2,2,0).
- psl_betweenEq(2,2,1).
- psl_betweenEq(2,2,2).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,0,2).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(0,1,2).
- tr_has_role(0,2,0).
- tr_has_role(0,2,1).
- tr_has_role(0,2,2).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,0,2).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).
- tr_has_role(1,1,2).
- tr_has_role(1,2,0).
- tr_has_role(1,2,1).
- tr_has_role(1,2,2).
- tr_has_role(2,0,0).
- tr_has_role(2,0,1).
- tr_has_role(2,0,2).
- tr_has_role(2,1,0).
- tr_has_role(2,1,1).
- tr_has_role(2,1,2).
- tr_has_role(2,2,0).
- tr_has_role(2,2,1).
- tr_has_role(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

B.2.2 \( H_{DOLCE}.participation \models T_j \)

\( H_{DOLCE}.participation \models H_{PSL}.participates \)

**Proof attempt for axiom 1 of PSL._participates:**

% Reading from file /stl/cchui/macleod-master qs/test/conversions/  
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t)))).
(all x (ED(x) -> (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t)))).
(all x all y all t (PC(x,y,t) <-> (ED(x) & PD(y) & T(t) & (all t1 ((P(t1,t)  
& T(t1)) -> PC(x,y,t1)))))).

% Reading from file /stl/cchui/macleod-master qs/test/conversions/  
dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t))))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t,t1,t2) -> PRE(x,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y))).
(all x all y (T(x) -> P(x,x))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & P(x,y) -> (exists z (T(z) & T(x) & T(z)))).
(all x all y (T(x) & T(y) & P(x,y) -> (exists z (T(z) & T(z) & T(x) & T(z)))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (T(z) & P(x,z) & P(y,z) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (T(z) & P(x,z) & P(y,z) & T(z)))).
(all x all y (T(x) & T(y) & AtP(x) -> T(x) & (all y (T(y) & P(y,x) -> y = x))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).
(all x (EV(x) | STV(x) -> PD(x))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).
(all x (R(x) -> AB(x))).
(all x (M(x) | F(x) | POB(x) -> PED(x))).
(all x (NPOB(x) -> NPED(x))).
(all x (ACH(x) | ACC(x) -> EV(x))).
(all x (ST(x) | PRO(x) -> STV(x))).
(all x (TL(x) -> TQ(x))).
(all x (SL(x) -> PQ(x))).
(all x (TR(x) | PR(x) | AR(x) -> R(x))).
(all x (APO(x) | NAPO(x) -> POB(x))).
(all x (MOB(x) | SOB(x) -> NPOB(x))).
(all x (T(x) -> TR(x))).
(all x (S(x) -> PR(x))).
(all x (ASO(x) | NASO(x) -> SOB(x))).
(all x (SAG(x) | SC(x) -> ASO(x))).
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).
(all x (ED(x) -> -PD(x) & -Q(x) & -AB(x))).
(all x (PD(x) -> -Q(x) & -AB(x))).
(all x (Q(x) -> -AB(x))).
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x))).
(all x (PED(x) -> -NPED(x) & -AS(x))).
(all x (NPED(x) -> -AS(x))).
(all x (PD(x) <-> EV(x) | STV(x))).
(all x (EV(x) -> -STV(x))).
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x))).
(all x (TQ(x) -> -PQ(x) & -AQ(x))).
(all x (PQ(x) -> -AQ(x))).
(all x (PED(x) <-> M(x) | F(x) | POB(x))).
(all x (M(x) -> -F(x) & -POB(x))).
(all x (F(x) -> -POB(x))).
(all x (EV(x) <-> ACH(x) | ACC(x))).
(all x (ACH(x) -> -ACC(x))).
(all x (STV(x) <-> ST(x) | PRO(x))).
(all x (ST(x) -> -PRO(x))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> -PR(x) & -AR(x))).
(all x (PR(x) -> -AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> -NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> -SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> -NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> -SC(x))).

%*****relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

%'no dolce_participation equivalent for tr_part_of'

%'no dolce_participation equivalent for tr_object but we are given the
additional info that an object is an endurant’)
%(all a (tr_object(a) -> ED(a))).

%'no dolce_participation equivalent for tr_active_in’

%'no dolce_participation equivalent for tr_passive_in’

%'no dolce_participation equivalent for tr_consumed_in’

%psl_2_Tr
%relevance mappings to requirements
%psl_2_Tr
%psl_prefix added to all terms to distinguish from other candidates

%'participates_in equivalent to tr_participates_in’
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

%'activity_occurrence equivalent to tr_event’
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

%'no psl equivalent for tr_has_role’

%'exists_at equivalent to tr_exists_at’
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

%'subactivity_occurrence equivalent to tr_part_of’
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).
%*********root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) \rightarrow (tr_participates_in(x,a,t) )).
(all a (tr_event(a) \rightarrow tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) \rightarrow (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) \rightarrow (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) \rightarrow tr_part_of(a_1,a_2))).
(all x (tr_object(x) \rightarrow tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) \rightarrow tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) \rightarrow tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) \rightarrow tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) \rightarrow g_Event(x))).
(all x (g_Object(x) \rightarrow g_Object(x))).
(all x all y (g_hasParticipant(x,y) \rightarrow g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) \rightarrow g_pre(x,y))).
(all y (g_Time(y) \rightarrow g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) \rightarrow psl_activity_occurrence(x))).
(all x (psl_object(x) \rightarrow psl_object(x))).
(all o all t (psl_is_occurring_at(o,t) \rightarrow psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) \rightarrow psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) \rightarrow psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) \rightarrow psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) \rightarrow psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) \rightarrow psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) \rightarrow psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) \rightarrow psl_betweenEq(t1,t2,t3))
).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
do    dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
do    dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
do    dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
do    dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
\[(\text{all} \ x \ (\text{TL}(x) \rightarrow \text{TL}(x))).\]
\[(\text{all} \ x \ (\text{SL}(x) \rightarrow \text{SL}(x))).\]
\[(\text{all} \ x \ (\text{TR}(x) \rightarrow \text{TR}(x))).\]
\[(\text{all} \ x \ (\text{PR}(x) \rightarrow \text{PR}(x))).\]
\[(\text{all} \ x \ (\text{AR}(x) \rightarrow \text{AR}(x))).\]
\[(\text{all} \ x \ (\text{APO}(x) \rightarrow \text{APO}(x))).\]
\[(\text{all} \ x \ (\text{NAPO}(x) \rightarrow \text{NAPO}(x))).\]
\[(\text{all} \ x \ (\text{MOB}(x) \rightarrow \text{MOB}(x))).\]
\[(\text{all} \ x \ (\text{SOB}(x) \rightarrow \text{SOB}(x))).\]

\[(\text{all} \ x \ (\text{S}(x) \rightarrow \text{S}(x))).\]
\[(\text{all} \ x \ (\text{ASO}(x) \rightarrow \text{ASO}(x))).\]
\[(\text{all} \ x \ (\text{NASO}(x) \rightarrow \text{NASO}(x))).\]
\[(\text{all} \ x \ (\text{SAG}(x) \rightarrow \text{SAG}(x))).\]
\[(\text{all} \ x \ (\text{SC}(x) \rightarrow \text{SC}(x))).\]
\[(\text{all} \ x \ (\text{PT}(x) \rightarrow \text{PT}(x))).\]
\[(\text{all} \ x \ (\text{PD}(x) \rightarrow \text{PD}(x))).\]

%goal
%ax1:
-((\text{all} \ x \ (\text{psl_activity_occurrence}(x) \rightarrow \neg \text{psl_object}(x) \& \neg \text{psl_timepoint}(x)))
  )).

\textbf{Counterexample found:}

% number = 1
% seconds = 0

% Interpretation of size 3

t = 0.

c1 = 0.

\text{psl\_beginof}(0) = 0.
\text{psl\_beginof}(1) = 0.
\text{psl\_beginof}(2) = 0.

\text{psl\_endof}(0) = 0.
\text{psl\_endof}(1) = 0.
\text{psl\_endof}(2) = 0.

f2(0) = 0.
f2(1) = 0.
\begin{align*}
  f_2(2) &= 0.
  
  f_3(0) &= 0. \
  f_3(1) &= 0. \
  f_3(2) &= 1.
  
  f_5(0) &= 1. \
  f_5(1) &= 0. \
  f_5(2) &= 1.
  
  f_{10}(0) &= 0. \
  f_{10}(1) &= 0. \
  f_{10}(2) &= 0.
  
  f_{1}(0,0) &= 0. \
  f_{1}(0,1) &= 2. \
  f_{1}(0,2) &= 0. \
  f_{1}(1,0) &= 0. \
  f_{1}(1,1) &= 0. \
  f_{1}(1,2) &= 0. \
  f_{1}(2,0) &= 0. \
  f_{1}(2,1) &= 0. \
  f_{1}(2,2) &= 0.
  
  f_{6}(0,0) &= 0. \
  f_{6}(0,1) &= 0. \
  f_{6}(0,2) &= 0. \
  f_{6}(1,0) &= 0. \
  f_{6}(1,1) &= 0. \
  f_{6}(1,2) &= 0. \
  f_{6}(2,0) &= 0. \
  f_{6}(2,1) &= 0. \
  f_{6}(2,2) &= 0.
  
  f_{7}(0,0) &= 0. \
  f_{7}(0,1) &= 0. \
  f_{7}(0,2) &= 0. \
  f_{7}(1,0) &= 0. \
  f_{7}(1,1) &= 0. \
  f_{7}(1,2) &= 0. \
  f_{7}(2,0) &= 0. \
  f_{7}(2,1) &= 0. \
  f_{7}(2,2) &= 0.
\end{align*}
\[ f_8(0,0) = 0. \]
\[ f_8(0,1) = 0. \]
\[ f_8(0,2) = 0. \]
\[ f_8(1,0) = 0. \]
\[ f_8(1,1) = 1. \]
\[ f_8(1,2) = 0. \]
\[ f_8(2,0) = 0. \]
\[ f_8(2,1) = 0. \]
\[ f_8(2,2) = 0. \]

\[ f_9(0,0) = 0. \]
\[ f_9(0,1) = 0. \]
\[ f_9(0,2) = 0. \]
\[ f_9(1,0) = 0. \]
\[ f_9(1,1) = 1. \]
\[ f_9(1,2) = 0. \]
\[ f_9(2,0) = 0. \]
\[ f_9(2,1) = 0. \]
\[ f_9(2,2) = 0. \]

\[ f_{11}(0,0) = 0. \]
\[ f_{11}(0,1) = 0. \]
\[ f_{11}(0,2) = 0. \]
\[ f_{11}(1,0) = 0. \]
\[ f_{11}(1,1) = 1. \]
\[ f_{11}(1,2) = 0. \]
\[ f_{11}(2,0) = 0. \]
\[ f_{11}(2,1) = 0. \]
\[ f_{11}(2,2) = 0. \]

\[ f_{12}(0,0) = 0. \]
\[ f_{12}(0,1) = 0. \]
\[ f_{12}(0,2) = 0. \]
\[ f_{12}(1,0) = 0. \]
\[ f_{12}(1,1) = 1. \]
\[ f_{12}(1,2) = 0. \]
\[ f_{12}(2,0) = 0. \]
\[ f_{12}(2,1) = 0. \]
\[ f_{12}(2,2) = 0. \]

\[ f_{12}(0,0,0) = 0. \]
\[ f_{12}(0,0,1) = 0. \]
\[ f_{4}(0,0,2) = 0. \]
\[ f_{4}(0,1,0) = 0. \]
\[ f_{4}(0,1,1) = 0. \]
\[ f_{4}(0,1,2) = 0. \]
\[ f_{4}(0,2,0) = 0. \]
\[ f_{4}(0,2,1) = 0. \]
\[ f_{4}(1,0,0) = 0. \]
\[ f_{4}(1,0,1) = 0. \]
\[ f_{4}(1,0,2) = 0. \]
\[ f_{4}(1,1,0) = 0. \]
\[ f_{4}(1,1,1) = 0. \]
\[ f_{4}(1,1,2) = 0. \]
\[ f_{4}(1,2,0) = 0. \]
\[ f_{4}(1,2,1) = 0. \]
\[ f_{4}(1,2,2) = 0. \]
\[ f_{4}(2,0,0) = 0. \]
\[ f_{4}(2,0,1) = 0. \]
\[ f_{4}(2,0,2) = 0. \]
\[ f_{4}(2,1,0) = 0. \]
\[ f_{4}(2,1,1) = 0. \]
\[ f_{4}(2,1,2) = 0. \]
\[ f_{4}(2,2,0) = 0. \]
\[ f_{4}(2,2,1) = 0. \]
\[ f_{4}(2,2,2) = 0. \]

\[ f_{13}(0,0,0) = 0. \]
\[ f_{13}(0,0,1) = 0. \]
\[ f_{13}(0,0,2) = 0. \]
\[ f_{13}(0,1,0) = 0. \]
\[ f_{13}(0,1,1) = 0. \]
\[ f_{13}(0,1,2) = 0. \]
\[ f_{13}(0,2,0) = 0. \]
\[ f_{13}(0,2,1) = 0. \]
\[ f_{13}(0,2,2) = 0. \]
\[ f_{13}(1,0,0) = 0. \]
\[ f_{13}(1,0,1) = 0. \]
\[ f_{13}(1,0,2) = 0. \]
\[ f_{13}(1,1,0) = 0. \]
\[ f_{13}(1,1,1) = 0. \]
\[ f_{13}(1,1,2) = 0. \]
\[ f_{13}(1,2,0) = 0. \]
\[ f_{13}(1,2,1) = 0. \]
\text{i3}(1,2,2) = 0.
\text{i3}(2,0,0) = 0.
\text{i3}(2,0,1) = 0.
\text{i3}(2,0,2) = 0.
\text{i3}(2,1,0) = 0.
\text{i3}(2,1,1) = 0.
\text{i3}(2,1,2) = 0.
\text{i3}(2,2,0) = 0.
\text{i3}(2,2,1) = 0.
\text{i3}(2,2,2) = 0.

- \text{AB}(0).
  \text{AB}(1).
  \text{AB}(2).

- \text{ACC}(0).
- \text{ACC}(1).
- \text{ACC}(2).

- \text{ACH}(0).
- \text{ACH}(1).
- \text{ACH}(2).

- \text{APO}(0).
- \text{APO}(1).
- \text{APO}(2).

- \text{AQ}(0).
- \text{AQ}(1).
- \text{AQ}(2).

- \text{AR}(0).
- \text{AR}(1).
- \text{AR}(2).

- \text{AS}(0).
- \text{AS}(1).
- \text{AS}(2).

- \text{ASO}(0).
- \text{ASO}(1).
- \text{ASO}(2).
- $\text{AtP}(0)$.
  $\text{AtP}(1)$.
  $\text{AtP}(2)$.

- $\text{ED}(0)$.
  $\text{ED}(1)$.
  $\text{ED}(2)$.

- $\text{EV}(0)$.
  $\text{EV}(1)$.
  $\text{EV}(2)$.

- $\text{F}(0)$.
  $\text{F}(1)$.
  $\text{F}(2)$.

- $\text{M}(0)$.
  $\text{M}(1)$.
  $\text{M}(2)$.

- $\text{MOB}(0)$.
  $\text{MOB}(1)$.
  $\text{MOB}(2)$.

- $\text{NAPO}(0)$.
  $\text{NAPO}(1)$.
  $\text{NAPO}(2)$.

- $\text{NASO}(0)$.
  $\text{NASO}(1)$.
  $\text{NASO}(2)$.

- $\text{NPED}(0)$.
  $\text{NPED}(1)$.
  $\text{NPED}(2)$.

- $\text{NPOB}(0)$.
  $\text{NPOB}(1)$.
  $\text{NPOB}(2)$.

- $\text{PD}(0)$.
  $\text{PD}(1)$.
  $\text{PD}(2)$. 
- PED(0).
- PED(1).
- PED(2).

- POB(0).
- POB(1).
- POB(2).

- PQ(0).
- PQ(1).
- PQ(2).

- PR(0).
- PR(1).
- PR(2).

- PRO(0).
- PRO(1).
- PRO(2).

- PT(0).
- PT(1).
- PT(2).

- Q(0).
- Q(1).
- Q(2).

- R(0).
- R(1).
- R(2).

- S(0).
- S(1).
- S(2).

- SAG(0).
- SAG(1).
- SAG(2).

- SC(0).
- SC(1).
- SC(2).
- SL(0).
- SL(1).
- SL(2).
- SOB(0).
- SOB(1).
- SOB(2).

ST(0).
- ST(1).
- ST(2).

STV(0).
- STV(1).
- STV(2).

- T(0).
  T(1).
  T(2).

- TL(0).
- TL(1).
- TL(2).

- TQ(0).
- TQ(1).
- TQ(2).

- TR(0).
  TR(1).
  TR(2).

  psl_activity_occurrence(0).
- psl_activity_occurrence(1).
- psl_activity_occurrence(2).

- psl_object(0).
- psl_object(1).
- psl_object(2).

  psl_timepoint(0).
- psl_timepoint(1).
- psl_timepoint(2).

  tr_event(0).
- tr_event(1).
- tr_event(2).

- DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).

- O(0,0).
- O(0,1).
- O(0,2).
- O(1,0).
  O(1,1).
- O(1,2).
- O(2,0).
- O(2,1).
- O(2,2).

- P(0,0).
- P(0,1).
- P(0,2).
- P(1,0).
  P(1,1).
- P(1,2).
- P(2,0).
- P(2,1).
- P(2,2).

- PP(0,0).
- PP(0,1).
- PP(0,2).
- PP(1,0).
- PP(1,1).
- PP(1,2).
- PP(2,0).
- PP(2,1).
- PP(2,2).
- PRE(0,0).
- PRE(0,1).
- PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- PRE(1,2).
- PRE(2,0).
- PRE(2,1).
- PRE(2,2).
- U(0,0).
- U(0,1).
- U(0,2).
- U(1,0).
- U(1,1).
- U(1,2).
- U(2,0).
- U(2,1).
- U(2,2).
- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(0,2).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).
- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(2,0).
- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).
- \text{tr\_exists\_at}(0,0).
  \text{tr\_exists\_at}(0,1).
- \text{tr\_exists\_at}(0,2).
- \text{tr\_exists\_at}(1,0).
- \text{tr\_exists\_at}(1,1).
- \text{tr\_exists\_at}(1,2).
- \text{tr\_exists\_at}(2,0).
  \text{tr\_exists\_at}(2,1).
- \text{tr\_exists\_at}(2,2).

- \text{tr\_part\_of}(0,0).
- \text{tr\_part\_of}(0,1).
- \text{tr\_part\_of}(0,2).
- \text{tr\_part\_of}(1,0).
- \text{tr\_part\_of}(1,1).
- \text{tr\_part\_of}(1,2).
- \text{tr\_part\_of}(2,0).
- \text{tr\_part\_of}(2,1).
- \text{tr\_part\_of}(2,2).

- \text{PC}(0,0,0).
- \text{PC}(0,0,1).
- \text{PC}(0,0,2).
- \text{PC}(0,1,0).
- \text{PC}(0,1,1).
- \text{PC}(0,1,2).
- \text{PC}(0,2,0).
- \text{PC}(0,2,1).
- \text{PC}(0,2,2).
- \text{PC}(1,0,0).
- \text{PC}(1,0,1).
- \text{PC}(1,0,2).
- \text{PC}(1,1,0).
- \text{PC}(1,1,1).
- \text{PC}(1,1,2).
- \text{PC}(1,2,0).
- \text{PC}(1,2,1).
- \text{PC}(1,2,2).
- \text{PC}(2,0,0).
  \text{PC}(2,0,1).
- \text{PC}(2,0,2).
- \text{PC}(2,1,0).
- pc(2,1,1).
- pc(2,1,2).
- pc(2,2,0).
- pc(2,2,1).
- pc(2,2,2).

- sum(0,0,0).
- sum(0,0,1).
- sum(0,0,2).
- sum(0,1,0).
- sum(0,1,1).
- sum(0,1,2).
- sum(0,2,0).
- sum(0,2,1).
- sum(0,2,2).
- sum(1,0,0).
- sum(1,0,1).
- sum(1,0,2).
- sum(1,1,0).
- sum(1,1,1).
- sum(1,1,2).
- sum(1,2,0).
- sum(1,2,1).
- sum(1,2,2).
- sum(2,0,0).
- sum(2,0,1).
- sum(2,0,2).
- sum(2,1,0).
- sum(2,1,1).
- sum(2,1,2).
- sum(2,2,0).
- sum(2,2,1).
- sum(2,2,2).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

\[ H_{DOLCE\_participation} \models H_{gangemi\_participation} \]

**Proof attempt for axiom 1 of gangemi_participation:**

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t)))).
(all x (ED(x) -> (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t))).
(all x all y all t (PC(x,y,t) <-> (ED(x) & PD(y) & T(t) & (all t1 ((P(t1,t) & T(t1)) -> PC(x,y,t1))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t)))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t,t1,t2) -> PRE(x,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> T(x) <-> T(y))).
(all x all y (T(x) -> P(x,y))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (T(z) & P(z,x) & -O(z,y)))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z)))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y)))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y)))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x)))
(all x (PED(x) | NPED(x) | AS(x) -> ED(x)))
(all x (EV(x) | STV(x) -> PD(x)))
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x)))
(all x (R(x) -> AB(x)))
(all x (M(x) | F(x) | POB(x) -> PED(x)))
(all x (NPOB(x) -> NPED(x)))
(all x (ACH(x) | ACC(x) -> EV(x)))
(all x (ST(x) | PRO(x) -> STV(x)))
(all x (TL(x) -> TQ(x)))
(all x (SL(x) -> PQ(x)))
(all x (TR(x) | PR(x) | AR(x) -> R(x)))
(all x (APO(x) | NAPO(x) -> POB(x)))
(all x (MOB(x) | SOB(x) -> NPOB(x)))
(all x (T(x) -> TR(x)))
(all x (S(x) -> PR(x)))
(all x (ASO(x) | NASO(x) -> SOB(x)))
(all x (SAG(x) | SC(x) -> ASO(x)))
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x)))
(all x (ED(x) -> -PD(x) & -Q(x) & -AB(x)))
(all x (PD(x) -> -Q(x) & -AB(x)))
(all x (Q(x) -> -AB(x)))
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x)))
(all x (PED(x) -> -NPED(x) & -AS(x)))
(all x (NPED(x) -> -AS(x)))
(all x (PD(x) <-> EV(x) | STV(x)))
(all x (EV(x) -> -STV(x)))
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x)))
(all x (TQ(x) -> -PQ(x) & -AQ(x)))
(all x (PQ(x) -> -AQ(x)))
(all x (PED(x) <-> M(x) | F(x) | POB(x)))
(all x (M(x) -> -F(x) & -POB(x)))
(all x (F(x) -> -POB(x)))
\[
\begin{align*}
\forall x \ (EV(x) \leftrightarrow ACH(x) \, | \, ACC(x))). \\
\forall x \ (ACH(x) \rightarrow -ACC(x))). \\
\forall x \ (STV(x) \leftrightarrow ST(x) \, | \, PRO(x))). \\
\forall x \ (ST(x) \rightarrow -PRO(x))). \\
\forall x \ (R(x) \leftrightarrow TR(x) \, | \, PR(x) \, | \, AR(x))). \\
\forall x \ (TR(x) \rightarrow -PR(x) \, \& \, -AR(x))). \\
\forall x \ (PR(x) \rightarrow -AR(x))). \\
\forall x \ (POB(x) \leftrightarrow APO(x) \, | \, NAPO(x))). \\
\forall x \ (APO(x) \rightarrow -NAPO(x))). \\
\forall x \ (NPOB(x) \leftrightarrow MOB(x) \, | \, SOB(x))). \\
\forall x \ (MOB(x) \rightarrow -SOB(x))). \\
\forall x \ (SOB(x) \leftrightarrow ASO(x) \, | \, NASO(x))). \\
\forall x \ (ASO(x) \rightarrow -NASO(x))). \\
\forall x \ (ASO(x) \leftrightarrow SAG(x) \, | \, SC(x))). \\
\forall x \ (SAG(x) \rightarrow -SC(x))).
\end{align*}
\]

%%%% relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(\forall x \ \forall a \ \forall t \ (PC(x,a,t) \leftrightarrow tr_participates_in(x,a,t))).

% 'PD equivalent to tr_event'
(\forall a \ (PD(a) \leftrightarrow tr_event(a))).

% 'no dolce_participation equivalent for tr_has_role'

% 'PRE equivalent to tr_exists_at'
(\forall a \ \forall t \ (PRE(a,t) \leftrightarrow tr_exists_at(a,t))).

% 'no dolce_participation equivalent for tr_part_of'

% 'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant')
(\forall a \ (tr_object(a) \rightarrow ED(a))).

% 'no dolce_participation equivalent for tr_active_in'

% 'no dolce_participation equivalent for tr_passive_in'

% 'no dolce_participation equivalent for tr_consumed_in'
%gangemi_2_Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t )))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%**********root theory
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
% psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%ax1
-((all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y)))))))
.

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

t = 0.

c1 = 0.
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[
\begin{align*}
\text{psl\_beginof}(0) & = 0. \\
\text{psl\_beginof}(1) & = 0. \\
\text{psl\_beginof}(2) & = 0. \\
\text{psl\_endof}(0) & = 0. \\
\text{psl\_endof}(1) & = 0. \\
\text{psl\_endof}(2) & = 0. \\
f2(0) & = 0. \\
f2(1) & = 0. \\
f2(2) & = 0. \\
f3(0) & = 0. \\
f3(1) & = 0. \\
f3(2) & = 1. \\
f5(0) & = 1. \\
f5(1) & = 0. \\
f5(2) & = 1. \\
f10(0) & = 0. \\
f10(1) & = 0. \\
f10(2) & = 0. \\
f1(0,0) & = 0. \\
f1(0,1) & = 2. \\
f1(0,2) & = 0. \\
f1(1,0) & = 0. \\
f1(1,1) & = 0. \\
f1(1,2) & = 0. \\
f1(2,0) & = 0. \\
f1(2,1) & = 0. \\
f1(2,2) & = 0. \\
f6(0,0) & = 0. \\
f6(0,1) & = 0. \\
f6(0,2) & = 0. \\
f6(1,0) & = 0. \\
f6(1,1) & = 0. \\
f6(1,2) & = 0. \\
f6(2,0) & = 0. \\
f6(2,1) & = 0. \\
f6(2,2) & = 0.
\end{align*}
\]
\( f_7(0,0) = 0. \)
\( f_7(0,1) = 0. \)
\( f_7(0,2) = 0. \)
\( f_7(1,0) = 0. \)
\( f_7(1,1) = 0. \)
\( f_7(1,2) = 0. \)
\( f_7(2,0) = 0. \)
\( f_7(2,1) = 0. \)
\( f_7(2,2) = 0. \)

\( f_8(0,0) = 0. \)
\( f_8(0,1) = 0. \)
\( f_8(0,2) = 0. \)
\( f_8(1,0) = 0. \)
\( f_8(1,1) = 1. \)
\( f_8(1,2) = 0. \)
\( f_8(2,0) = 0. \)
\( f_8(2,1) = 0. \)
\( f_8(2,2) = 0. \)

\( f_9(0,0) = 0. \)
\( f_9(0,1) = 0. \)
\( f_9(0,2) = 0. \)
\( f_9(1,0) = 0. \)
\( f_9(1,1) = 1. \)
\( f_9(1,2) = 0. \)
\( f_9(2,0) = 0. \)
\( f_9(2,1) = 0. \)
\( f_9(2,2) = 0. \)

\( f_{11}(0,0) = 0. \)
\( f_{11}(0,1) = 0. \)
\( f_{11}(0,2) = 0. \)
\( f_{11}(1,0) = 0. \)
\( f_{11}(1,1) = 1. \)
\( f_{11}(1,2) = 0. \)
\( f_{11}(2,0) = 0. \)
\( f_{11}(2,1) = 0. \)
\( f_{11}(2,2) = 0. \)

\( f_{12}(0,0) = 0. \)
\( f_{12}(0,1) = 0. \)
\[ f_{12}(0,2) = 0. \]
\[ f_{12}(1,0) = 0. \]
\[ f_{12}(1,1) = 1. \]
\[ f_{12}(1,2) = 0. \]
\[ f_{12}(2,0) = 0. \]
\[ f_{12}(2,1) = 0. \]
\[ f_{12}(2,2) = 0. \]

\[ f_{14}(0,0) = 0. \]
\[ f_{14}(0,1) = 0. \]
\[ f_{14}(0,2) = 1. \]
\[ f_{14}(1,0) = 0. \]
\[ f_{14}(1,1) = 0. \]
\[ f_{14}(1,2) = 0. \]
\[ f_{14}(2,0) = 0. \]
\[ f_{14}(2,1) = 0. \]
\[ f_{14}(2,2) = 0. \]

\[ f_{4}(0,0,0) = 0. \]
\[ f_{4}(0,0,1) = 0. \]
\[ f_{4}(0,0,2) = 0. \]
\[ f_{4}(0,1,0) = 0. \]
\[ f_{4}(0,1,1) = 0. \]
\[ f_{4}(0,1,2) = 0. \]
\[ f_{4}(0,2,0) = 0. \]
\[ f_{4}(0,2,1) = 0. \]
\[ f_{4}(0,2,2) = 0. \]
\[ f_{4}(1,0,0) = 0. \]
\[ f_{4}(1,0,1) = 0. \]
\[ f_{4}(1,0,2) = 0. \]
\[ f_{4}(1,1,0) = 0. \]
\[ f_{4}(1,1,1) = 0. \]
\[ f_{4}(1,1,2) = 0. \]
\[ f_{4}(1,2,0) = 0. \]
\[ f_{4}(1,2,1) = 0. \]
\[ f_{4}(1,2,2) = 0. \]
\[ f_{4}(2,0,0) = 0. \]
\[ f_{4}(2,0,1) = 0. \]
\[ f_{4}(2,0,2) = 0. \]
\[ f_{4}(2,1,0) = 0. \]
\[ f_{4}(2,1,1) = 0. \]
\[ f_{4}(2,1,2) = 0. \]
\[ f_{4}(2,2,0) = 0. \]
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[ f_{4}(2, 2, 1) = 0. \]
\[ f_{4}(2, 2, 2) = 0. \]

\[ f_{13}(0, 0, 0) = 0. \]
\[ f_{13}(0, 0, 1) = 0. \]
\[ f_{13}(0, 0, 2) = 0. \]
\[ f_{13}(0, 1, 0) = 0. \]
\[ f_{13}(0, 1, 1) = 0. \]
\[ f_{13}(0, 1, 2) = 0. \]
\[ f_{13}(0, 2, 0) = 0. \]
\[ f_{13}(0, 2, 1) = 0. \]
\[ f_{13}(0, 2, 2) = 0. \]
\[ f_{13}(1, 0, 0) = 0. \]
\[ f_{13}(1, 0, 1) = 0. \]
\[ f_{13}(1, 0, 2) = 0. \]
\[ f_{13}(1, 1, 0) = 0. \]
\[ f_{13}(1, 1, 1) = 0. \]
\[ f_{13}(1, 1, 2) = 0. \]
\[ f_{13}(1, 2, 0) = 0. \]
\[ f_{13}(1, 2, 1) = 0. \]
\[ f_{13}(1, 2, 2) = 0. \]
\[ f_{13}(2, 0, 0) = 0. \]
\[ f_{13}(2, 0, 1) = 0. \]
\[ f_{13}(2, 0, 2) = 0. \]
\[ f_{13}(2, 1, 0) = 0. \]
\[ f_{13}(2, 1, 1) = 0. \]
\[ f_{13}(2, 1, 2) = 0. \]
\[ f_{13}(2, 2, 0) = 0. \]
\[ f_{13}(2, 2, 1) = 0. \]
\[ f_{13}(2, 2, 2) = 0. \]

- AB(0).
  - AB(1).
- AB(2).

- ACC(0).
- ACC(1).
- ACC(2).

- ACH(0).
- ACH(1).
- ACH(2).
Appendix B. Comparing Participation Ontologies: Proof Files

- APO(0).
- APO(1).
- APO(2).

- AQ(0).
- AQ(1).
- AQ(2).

- AR(0).
- AR(1).
- AR(2).

- AS(0).
- AS(1).
- AS(2).

- ASO(0).
- ASO(1).
- ASO(2).

- AtP(0).
  - AtP(1).
  - AtP(2).

- ED(0).
- ED(1).
  - ED(2).

- EV(0).
- EV(1).
- EV(2).

- F(0).
- F(1).
- F(2).

- M(0).
- M(1).
- M(2).

- MOB(0).
- MOB(1).
- MOB(2).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- NAPO(0).
- NAPO(1).
- NAPO(2).

- NASO(0).
- NASO(1).
- NASO(2).

- NPED(0).
- NPED(1).
- NPED(2).

- NPOB(0).
- NPOB(1).
- NPOB(2).

- PD(0).
- PD(1).
- PD(2).

- PED(0).
- PED(1).
- PED(2).

- POB(0).
- POB(1).
- POB(2).

- PQ(0).
- PQ(1).
- PQ(2).

- PR(0).
- PR(1).
- PR(2).

- PRO(0).
- PRO(1).
- PRO(2).

PT(0).
PT(1).
Appendix B. Comparing Participation Ontologies: Proof Files

PT(2).
- Q(0).
- Q(1).
- Q(2).
- R(0).
- R(1).
- R(2).
- S(0).
- S(1).
- S(2).
- SAG(0).
- SAG(1).
- SAG(2).
- SC(0).
- SC(1).
- SC(2).
- SL(0).
- SL(1).
- SL(2).
- SOB(0).
- SOB(1).
- SOB(2).
- ST(0).
- ST(1).
- ST(2).
- STV(0).
- STV(1).
- STV(2).
- T(0).
- T(1).
- T(2).
- TL(0).
- TL(1).
- TL(2).

- TQ(0).
- TQ(1).
- TQ(2).

- TR(0).
  TR(1).
- TR(2).

  g_Event(0).
- g_Event(1).
- g_Event(2).

- g_Object(0).
- g_Object(1).
- g_Object(2).

  tr_event(0).
- tr_event(1).
- tr_event(2).

- tr_object(0).
- tr_object(1).
- tr_object(2).

- DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).

- O(0,0).
- O(0,1).
- O(0,2).
- O(1,0).
  O(1,1).
- O(1,2).
- O(2,0).
- O(2,1).
- O(2,2).

- P(0,0).
- P(0,1).
- P(0,2).
- P(1,0).
  P(1,1).
- P(1,2).
- P(2,0).
- P(2,1).
- P(2,2).

- PP(0,0).
- PP(0,1).
- PP(0,2).
- PP(1,0).
  PP(1,1).
  PP(1,2).
  PP(2,0).
- PP(2,1).
- PP(2,2).

- PRE(0,0).
  PRE(0,1).
- PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- PRE(1,2).
- PRE(2,0).
  PRE(2,1).
- PRE(2,2).

- U(0,0).
- U(0,1).
- U(0,2).
- U(1,0).
  U(1,1).
- U(1,2).
- U(2,0).
- U(2,1).
- U(2,2).
- g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(0,2).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).
- g_hasParticipant(1,2).
- g_hasParticipant(2,0).
- g_hasParticipant(2,1).
- g_hasParticipant(2,2).

- g_pre(0,0).
- g_pre(0,1).
- g_pre(0,2).
- g_pre(1,0).
- g_pre(1,1).
- g_pre(1,2).
- g_pre(2,0).
- g_pre(2,1).
- g_pre(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,0,2).
- PC(0,1,0).
- PC(0,1,1).
- PC(0,1,2).
- PC(0,2,0).
- PC(0,2,1).
- PC(0,2,2).
- PC(1,0,0).
- PC(1,0,1).
- PC(1,0,2).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- PC(1,1,0).
- PC(1,1,1).
- PC(1,1,2).
- PC(1,2,0).
- PC(1,2,1).
- PC(1,2,2).
- PC(2,0,0).
- PC(2,0,1).
- PC(2,0,2).
- PC(2,1,0).
- PC(2,1,1).
- PC(2,1,2).
- PC(2,2,0).
- PC(2,2,1).
- PC(2,2,2).

- SUM(0,0,0).
- SUM(0,0,1).
- SUM(0,0,2).
- SUM(0,1,0).
- SUM(0,1,1).
- SUM(0,1,2).
- SUM(0,2,0).
- SUM(0,2,1).
- SUM(0,2,2).
- SUM(1,0,0).
- SUM(1,0,1).
- SUM(1,0,2).
- SUM(1,1,0).
- SUM(1,1,1).
- SUM(1,1,2).
- SUM(1,2,0).
- SUM(1,2,1).
- SUM(1,2,2).
- SUM(2,0,0).
- SUM(2,0,1).
- SUM(2,0,2).
- SUM(2,1,0).
- SUM(2,1,1).
- SUM(2,1,2).
- SUM(2,2,0).
- SUM(2,2,1).
- SUM(2,2,2).
- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

\[ H\_DOLCE\_participation \models H\_T_R \]

**Proof attempt for CQ1 of \( T_R \):**

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolce_participation.p9
(dolce_participation.p9)
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t)))).
(all x (ED(x) -> (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t)))).
(all x all y all t (PC(x,y,t) <-> (ED(x) & PD(y) & T(t) & (all t1 ((P(t1,t) & T(t1)) -> PC(x,y,t1)))))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t)))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t,t1,t2) -> PRE(x,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y)))).
(all x all y (T(x) -> P(x,x))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (T(z) & P(z,x) & -O(z,y))))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y))))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z)))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y)))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y)))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolcetaxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).
(all x (EV(x) | STV(x) -> PD(x))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).
(all x (R(x) -> AB(x))).
(all x (M(x) | F(x) | POB(x) -> PED(x))).
(all x (NPOB(x) -> NPED(x))).
(all x (ACH(x) | ACC(x) -> EV(x))).
(all x (ST(x) | PRO(x) -> STV(x))).
(all x (TL(x) -> TQ(x))).
(all x (SL(x) -> PQ(x))).
(all x (TR(x) | PR(x) | AR(x) -> R(x))).
(all x (APO(x) | NAPO(x) -> POB(x))).
(all x (MOB(x) | SOB(x) -> NPOB(x))).
(all x (T(x) -> TR(x))).
(all x (S(x) -> PR(x))).
(all x (ASO(x) | NASO(x) -> SOB(x))).
(all x (SAG(x) | SC(x) -> ASO(x))).
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).
(all x (ED(x) -> -PD(x) & -Q(x) & -AB(x))).
(all x (PD(x) -> -Q(x) & -AB(x))).
(all x (Q(x) -> -AB(x))).
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x))).
(all x (PED(x) -> -NPED(x) & -AS(x))).
(all x (NPED(x) -> -AS(x))).
(all x (PD(x) <-> EV(x) | STV(x))).
(all x (EV(x) -> -STV(x))).
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x))).
(all x (TQ(x) -> -PQ(x) & -AQ(x))).
(all x (PQ(x) -> -AQ(x))).
(all x (PED(x) <-> M(x) | F(x) | POB(x))).
(all x (M(x) -> -F(x) & -POB(x))).
(all x (F(x) -> -POB(x))).
(all x (EV(x) <-> ACH(x) | ACC(x))).
(all x (ACH(x) -> -ACC(x))).
(all x (STV(x) <-> ST(x) | PRO(x))).
(all x (ST(x) -> -PRO(x))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> -PR(x) & -AR(x))).
(all x (PR(x) -> -AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> -NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> -SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> -NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> -SC(x))).

%-------relevance mappings to requirements
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

% dolce_2_tr
% katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

% 'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

% 'no dolce_participation equivalent for tr_has_role'

% 'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

% 'no dolce_participation equivalent for tr_part_of'

% 'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant'
%(all a (tr_object(a) -> ED(a))).

% 'no dolce_participation equivalent for tr_active_in'

% 'no dolce_participation equivalent for tr_passive_in'

% 'no dolce_participation equivalent for tr_consumed_in'

%***********root theory
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

% gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

% psl_participation.p9 signature
% psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

% dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
% dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t2 (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
% dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
% dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
\[
\begin{align*}
&\quad (\text{all } x \ (\text{AtP}(x) \rightarrow \text{AtP}(x))). \\
&\quad (\text{all } x \ \text{all } y \ (\text{U}(x,y) \rightarrow \text{U}(x,y))). \\
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
&\quad \text{dolce_taxonomy.p9} \\
&\quad (\text{all } x \ (\text{AB}(x) \rightarrow \text{AB}(x))). \\
&\quad (\text{all } x \ (\text{PED}(x) \rightarrow \text{PED}(x))). \\
&\quad (\text{all } x \ (\text{NPED}(x) \rightarrow \text{NPED}(x))). \\
&\quad (\text{all } x \ (\text{AS}(x) \rightarrow \text{AS}(x))). \\
&\quad (\text{all } x \ (\text{EV}(x) \rightarrow \text{EV}(x))). \\
&\quad (\text{all } x \ (\text{STV}(x) \rightarrow \text{STV}(x))). \\
&\quad (\text{all } x \ (\text{TQ}(x) \rightarrow \text{TQ}(x))). \\
&\quad (\text{all } x \ (\text{FQ}(x) \rightarrow \text{FQ}(x))). \\
&\quad (\text{all } x \ (\text{AQ}(x) \rightarrow \text{AQ}(x))). \\
&\quad (\text{all } x \ (\text{R}(x) \rightarrow \text{R}(x))). \\
&\quad (\text{all } x \ (\text{M}(x) \rightarrow \text{M}(x))). \\
&\quad (\text{all } x \ (\text{F}(x) \rightarrow \text{F}(x))). \\
&\quad (\text{all } x \ (\text{POB}(x) \rightarrow \text{POB}(x))). \\
&\quad (\text{all } x \ (\text{NPOB}(x) \rightarrow \text{NPOB}(x))). \\
&\quad (\text{all } x \ (\text{ACH}(x) \rightarrow \text{ACH}(x))). \\
&\quad (\text{all } x \ (\text{ACC}(x) \rightarrow \text{ACC}(x))). \\
&\quad (\text{all } x \ (\text{ST}(x) \rightarrow \text{ST}(x))). \\
&\quad (\text{all } x \ (\text{PRO}(x) \rightarrow \text{PRO}(x))). \\
&\quad (\text{all } x \ (\text{TL}(x) \rightarrow \text{TL}(x))). \\
&\quad (\text{all } x \ (\text{SL}(x) \rightarrow \text{SL}(x))). \\
&\quad (\text{all } x \ (\text{TR}(x) \rightarrow \text{TR}(x))). \\
&\quad (\text{all } x \ (\text{PR}(x) \rightarrow \text{PR}(x))). \\
&\quad (\text{all } x \ (\text{AR}(x) \rightarrow \text{AR}(x))). \\
&\quad (\text{all } x \ (\text{APO}(x) \rightarrow \text{APO}(x))). \\
&\quad (\text{all } x \ (\text{NAPO}(x) \rightarrow \text{NAPO}(x))). \\
&\quad (\text{all } x \ (\text{MOB}(x) \rightarrow \text{MOB}(x))). \\
&\quad (\text{all } x \ (\text{SOB}(x) \rightarrow \text{SOB}(x))). \\
&\quad (\text{all } x \ (\text{S}(x) \rightarrow \text{S}(x))). \\
&\quad (\text{all } x \ (\text{ASO}(x) \rightarrow \text{ASO}(x))). \\
&\quad (\text{all } x \ (\text{NASO}(x) \rightarrow \text{NASO}(x))). \\
&\quad (\text{all } x \ (\text{SAG}(x) \rightarrow \text{SAG}(x))). \\
&\quad (\text{all } x \ (\text{SC}(x) \rightarrow \text{SC}(x))). \\
&\quad (\text{all } x \ (\text{PT}(x) \rightarrow \text{PT}(x))). \\
&\quad (\text{all } x \ (\text{PD}(x) \rightarrow \text{PD}(x))). \\
&\quad \% goal
\end{align*}
\]
% CQL 'Anything that participates in some event has some role when they are participating.'
-((all x all a all t
   (tr_participates_in(x,a,t)
    & tr_event(a)
    ->
     (exists z
      tr_has_role(x,z,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

c1 = 0.
c2 = 1.
c3 = 2.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.
psl_endof(2) = 0.

f2(0) = 1.
f2(1) = 0.
f2(2) = 0.

f3(0) = 2.
f3(1) = 0.
f3(2) = 0.

f5(0) = 2.
f5(1) = 2.
f5(2) = 0.

f10(0) = 0.
f10(1) = 0.
f_{10}(2) = 0.

f_{1}(0,0) = 0.
f_{1}(0,1) = 0.
f_{1}(0,2) = 0.
f_{1}(1,0) = 0.
f_{1}(1,1) = 0.
f_{1}(1,2) = 0.
f_{1}(2,0) = 0.
f_{1}(2,1) = 0.
f_{1}(2,2) = 0.

f_{6}(0,0) = 0.
f_{6}(0,1) = 0.
f_{6}(0,2) = 0.
f_{6}(1,0) = 0.
f_{6}(1,1) = 0.
f_{6}(1,2) = 0.
f_{6}(2,0) = 0.
f_{6}(2,1) = 0.
f_{6}(2,2) = 0.

f_{7}(0,0) = 0.
f_{7}(0,1) = 0.
f_{7}(0,2) = 0.
f_{7}(1,0) = 0.
f_{7}(1,1) = 0.
f_{7}(1,2) = 0.
f_{7}(2,0) = 0.
f_{7}(2,1) = 0.
f_{7}(2,2) = 0.

f_{8}(0,0) = 0.
f_{8}(0,1) = 0.
f_{8}(0,2) = 0.
f_{8}(1,0) = 0.
f_{8}(1,1) = 0.
f_{8}(1,2) = 0.
f_{8}(2,0) = 0.
f_{8}(2,1) = 0.
f_{8}(2,2) = 2.

f_{9}(0,0) = 0.
f9(0,1) = 0.
f9(0,2) = 0.
f9(1,0) = 0.
f9(1,1) = 0.
f9(1,2) = 0.
f9(2,0) = 0.
f9(2,1) = 0.
f9(2,2) = 2.

f11(0,0) = 0.
f11(0,1) = 0.
f11(0,2) = 0.
f11(1,0) = 0.
f11(1,1) = 0.
f11(1,2) = 0.
f11(2,0) = 0.
f11(2,1) = 0.
f11(2,2) = 2.

f12(0,0) = 0.
f12(0,1) = 0.
f12(0,2) = 0.
f12(1,0) = 0.
f12(1,1) = 0.
f12(1,2) = 0.
f12(2,0) = 0.
f12(2,1) = 0.
f12(2,2) = 2.

f4(0,0,0) = 0.
f4(0,0,1) = 0.
f4(0,0,2) = 0.
f4(0,1,0) = 0.
f4(0,1,1) = 0.
f4(0,1,2) = 0.
f4(0,2,0) = 0.
f4(0,2,1) = 0.
f4(0,2,2) = 0.
f4(1,0,0) = 0.
f4(1,0,1) = 0.
f4(1,0,2) = 0.
f4(1,1,0) = 0.
f4(1,1,1) = 0.
\[ f_4(1,1,2) = 0. \]
\[ f_4(1,2,0) = 0. \]
\[ f_4(1,2,1) = 0. \]
\[ f_4(1,2,2) = 0. \]
\[ f_4(2,0,0) = 0. \]
\[ f_4(2,0,1) = 0. \]
\[ f_4(2,0,2) = 0. \]
\[ f_4(2,1,0) = 0. \]
\[ f_4(2,1,1) = 0. \]
\[ f_4(2,1,2) = 0. \]
\[ f_4(2,2,0) = 0. \]
\[ f_4(2,2,1) = 0. \]
\[ f_4(2,2,2) = 0. \]

\[ f_{13}(0,0,0) = 0. \]
\[ f_{13}(0,0,1) = 0. \]
\[ f_{13}(0,0,2) = 0. \]
\[ f_{13}(0,1,0) = 0. \]
\[ f_{13}(0,1,1) = 0. \]
\[ f_{13}(0,1,2) = 0. \]
\[ f_{13}(0,2,0) = 0. \]
\[ f_{13}(0,2,1) = 0. \]
\[ f_{13}(0,2,2) = 0. \]
\[ f_{13}(1,0,0) = 0. \]
\[ f_{13}(1,0,1) = 0. \]
\[ f_{13}(1,0,2) = 0. \]
\[ f_{13}(1,1,0) = 0. \]
\[ f_{13}(1,1,1) = 0. \]
\[ f_{13}(1,1,2) = 0. \]
\[ f_{13}(1,2,0) = 0. \]
\[ f_{13}(1,2,1) = 0. \]
\[ f_{13}(1,2,2) = 0. \]
\[ f_{13}(2,0,0) = 0. \]
\[ f_{13}(2,0,1) = 0. \]
\[ f_{13}(2,0,2) = 0. \]
\[ f_{13}(2,1,0) = 0. \]
\[ f_{13}(2,1,1) = 0. \]
\[ f_{13}(2,1,2) = 0. \]
\[ f_{13}(2,2,0) = 0. \]
\[ f_{13}(2,2,1) = 0. \]
\[ f_{13}(2,2,2) = 0. \]

- \( AB(0) \).
- AB(1).
  AB(2).

- ACC(0).
- ACC(1).
- ACC(2).

- ACH(0).
- ACH(1).
- ACH(2).

- APO(0).
- APO(1).
- APO(2).

- AQ(0).
- AQ(1).
- AQ(2).

- AR(0).
- AR(1).
- AR(2).

- AS(0).
- AS(1).
- AS(2).

- ASO(0).
- ASO(1).
- ASO(2).

- AtP(0).
- AtP(1).
  AtP(2).

   ED(0).
- ED(1).
- ED(2).

- EV(0).
- EV(1).
- EV(2).
- F(0).
- F(1).
- F(2).

- M(0).
- M(1).
- M(2).

- MOB(0).
- MOB(1).
- MOB(2).

- NAPO(0).
- NAPO(1).
- NAPO(2).

- NASO(0).
- NASO(1).
- NASO(2).

- NPED(0).
- NPED(1).
- NPED(2).

- NPOB(0).
- NPOB(1).
- NPOB(2).

- PD(0).
  PD(1).
  PD(2).

- PED(0).
  PED(1).
  PED(2).

- POB(0).
  POB(1).
  POB(2).

- PQ(0).
  PQ(1).
  PQ(2).
- PR(0).
- PR(1).
- PR(2).

- PRO(0).
- PRO(1).
- PRO(2).

- PT(0).
- PT(1).
- PT(2).

- Q(0).
- Q(1).
- Q(2).

- R(0).
- R(1).
- R(2).

- S(0).
- S(1).
- S(2).

- SAG(0).
- SAG(1).
- SAG(2).

- SC(0).
- SC(1).
- SC(2).

- SL(0).
- SL(1).
- SL(2).

- SOB(0).
- SOB(1).
- SOB(2).

- ST(0).
- ST(1).
- \text{ST}(2).
- \text{STV}(0).
  \text{STV}(1).
- \text{STV}(2).
- \text{T}(0).
- \text{T}(1).
- \text{T}(2).
- \text{TL}(0).
- \text{TL}(1).
- \text{TL}(2).
- \text{TQ}(0).
- \text{TQ}(1).
- \text{TQ}(2).
- \text{TR}(0).
- \text{TR}(1).
- \text{TR}(2).
- \text{tr}_\text{event}(0).
  \text{tr}_\text{event}(1).
- \text{tr}_\text{event}(2).
- \text{DJ}(0,0).
- \text{DJ}(0,1).
- \text{DJ}(0,2).
- \text{DJ}(1,0).
- \text{DJ}(1,1).
- \text{DJ}(1,2).
- \text{DJ}(2,0).
- \text{DJ}(2,1).
- \text{DJ}(2,2).
- \text{O}(0,0).
- \text{O}(0,1).
- \text{O}(0,2).
- \text{O}(1,0).
- \text{O}(1,1).
- \text{O}(1,2).
- \text{O}(2,0).
Appendix B. Comparing Participation Ontologies: Proof Files

- $O(0,0)$.
  - $O(0,1)$.
  - $O(0,2)$.
  - $O(0,3)$.
  - $O(1,0)$.
  - $O(1,1)$.
  - $O(1,2)$.
  - $O(1,3)$.
  - $O(2,0)$.
  - $O(2,1)$.
  - $O(2,2)$.

- $P(0,0)$.
  - $P(0,1)$.
  - $P(0,2)$.
  - $P(0,3)$.
  - $P(1,0)$.
  - $P(1,1)$.
  - $P(1,2)$.
  - $P(1,3)$.
  - $P(2,0)$.
  - $P(2,1)$.
  - $P(2,2)$.
  - $P(2,3)$.

- $PP(0,0)$.
  - $PP(0,1)$.
  - $PP(0,2)$.
  - $PP(0,3)$.
  - $PP(1,0)$.
  - $PP(1,1)$.
  - $PP(1,2)$.
  - $PP(1,3)$.
  - $PP(2,0)$.
  - $PP(2,1)$.
  - $PP(2,2)$.

- $PRE(0,0)$.
  - $PRE(0,1)$.
    - $PRE(0,2)$.
  - $PRE(1,0)$.
    - $PRE(1,1)$.
    - $PRE(1,2)$.
  - $PRE(2,0)$.
    - $PRE(2,1)$.
    - $PRE(2,2)$.

- $U(0,0)$.
  - $U(0,1)$.
  - $U(0,2)$.
  - $U(1,0)$.
  - $U(1,1)$.
  - $U(1,2)$.
  - $U(2,0)$.
  - $U(2,1)$.
  - $U(2,2)$. 
- \texttt{tr_exists_at(0,0)}.
- \texttt{tr_exists_at(0,1)}.
  \quad \texttt{tr_exists_at(0,2)}.
- \texttt{tr_exists_at(1,0)}.
- \texttt{tr_exists_at(1,1)}.
  \quad \texttt{tr_exists_at(1,2)}.
- \texttt{tr_exists_at(2,0)}.
- \texttt{tr_exists_at(2,1)}.
- \texttt{tr_exists_at(2,2)}.

- \texttt{PC(0,0,0)}.
- \texttt{PC(0,0,1)}.
- \texttt{PC(0,0,2)}.
- \texttt{PC(0,1,0)}.
- \texttt{PC(0,1,1)}.
  \quad \texttt{PC(0,1,2)}.
- \texttt{PC(0,2,0)}.
- \texttt{PC(0,2,1)}.
- \texttt{PC(0,2,2)}.
- \texttt{PC(1,0,0)}.
- \texttt{PC(1,0,1)}.
- \texttt{PC(1,0,2)}.
- \texttt{PC(1,1,0)}.
- \texttt{PC(1,1,1)}.
- \texttt{PC(1,1,2)}.
- \texttt{PC(1,2,0)}.
- \texttt{PC(1,2,1)}.
- \texttt{PC(1,2,2)}.
- \texttt{PC(2,0,0)}.
- \texttt{PC(2,0,1)}.
- \texttt{PC(2,0,2)}.
- \texttt{PC(2,1,0)}.
- \texttt{PC(2,1,1)}.
- \texttt{PC(2,1,2)}.
- \texttt{PC(2,2,0)}.
- \texttt{PC(2,2,1)}.
- \texttt{PC(2,2,2)}.

- \texttt{SUM(0,0,0)}.
- \texttt{SUM(0,0,1)}.
- \texttt{SUM(0,0,2)}.
- \texttt{SUM(0,1,0)}.
- \texttt{SUM(0,1,1)}.
- SUM(0,1,2).
- SUM(0,2,0).
- SUM(0,2,1).
- SUM(0,2,2).
- SUM(1,0,0).
- SUM(1,0,1).
- SUM(1,0,2).
- SUM(1,1,0).
- SUM(1,1,1).
- SUM(1,1,2).
- SUM(1,2,0).
- SUM(1,2,1).
- SUM(1,2,2).
- SUM(2,0,0).
- SUM(2,0,1).
- SUM(2,0,2).
- SUM(2,1,0).
- SUM(2,1,1).
- SUM(2,1,2).
- SUM(2,2,0).
- SUM(2,2,1).
- SUM(2,2,2).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,0,2).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(0,1,2).
- tr_has_role(0,2,0).
- tr_has_role(0,2,1).
- tr_has_role(0,2,2).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,0,2).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).
- tr_has_role(1,1,2).
- tr_has_role(1,2,0).
- tr_has_role(1,2,1).
- tr_has_role(1,2,2).
- tr_has_role(2,0,0).
- tr_has_role(2,0,1).
B.2.3  $H_{\text{gangemi}\_\text{participation}} \models T_j$

$H_{\text{gangemi}\_\text{participation}} \models H_{\text{DOLCE}\_\text{participation}}$

Proof attempt for axiom 1 of DOLCE\_participation:

%common hierarchy
% Reading from file /stl/cchi/macleod-master/qs/test/conversions/gangemi.p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_object(x) -> (exists y g_pre(x,y))))).
(all x all y (g_pre(x,y) -> (g_Object(x) | g_Event(x)) & g_Time(y))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%dolce_2_Tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs

% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

% 'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

% 'no dolce_participation equivalent for tr_has_role'

% 'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

% 'no dolce_participation equivalent for tr_part_of'

% 'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant')
%(all a (tr_object(a) -> ED(a))).

% 'no dolce_participation equivalent for tr_active_in'

%'no dolce_participation equivalent for tr_passive_in’

%'no dolce_participation equivalent for tr_consumed_in’

%root theory:%Root Theory (collection of candidates’ signature term
tautologies)
% katsumi-thesis-searchcase-semantic-regts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))
  )).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))
  )).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))
  )).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))
  )).
(all x (tr_object(x) -> tr_object(x))
  )).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))
  )).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))
  )).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))
  )).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))
  )).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
% psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t)))
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x
  ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))
  )).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
    dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
    dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
    dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
    dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x)));

%goal
%ax1
-((all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).

  Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.

c1 = 0.

c2 = 0.

c3 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
   f1(1) = 0.

f2(0) = 0.
   f2(1) = 0.

f3(0) = 0.
   f3(1) = 0.

f4(0) = 0.
   f4(1) = 0.

f5(0,0) = 0.
   f5(0,1) = 0.
   f5(1,0) = 0.
   f5(1,1) = 0.

- ED(0).
- ED(1).

- PD(0).
- PD(1).

- T(0).
- T(1).

- g_Event(0).
- g_Event(1).

- g_Object(0).
- g_Object(1).

- g_Time(0).
- g_Time(1).

- tr_event(0).
- tr_event(1).

- tr_object(0).
- tr_object(1).
\[\begin{align*}
&\text{PRE}(0,0). \\
&\text{PRE}(0,1). \\
&\text{PRE}(1,0). \\
&\text{PRE}(1,1). \\
&\text{g\_hasParticipant}(0,0). \\
&\text{g\_hasParticipant}(0,1). \\
&\text{g\_hasParticipant}(1,0). \\
&\text{g\_hasParticipant}(1,1). \\
&\text{g\_pre}(0,0). \\
&\text{g\_pre}(0,1). \\
&\text{g\_pre}(1,0). \\
&\text{g\_pre}(1,1). \\
&\text{tr\_exists\_at}(0,0). \\
&\text{tr\_exists\_at}(0,1). \\
&\text{tr\_exists\_at}(1,0). \\
&\text{tr\_exists\_at}(1,1). \\
&\text{PC}(0,0,0). \\
&\text{PC}(0,0,1). \\
&\text{PC}(0,1,0). \\
&\text{PC}(0,1,1). \\
&\text{PC}(1,0,0). \\
&\text{PC}(1,0,1). \\
&\text{PC}(1,1,0). \\
&\text{PC}(1,1,1). \\
&\text{tr\_participates\_in}(0,0,0). \\
&\text{tr\_participates\_in}(0,0,1). \\
&\text{tr\_participates\_in}(0,1,0). \\
&\text{tr\_participates\_in}(0,1,1). \\
&\text{tr\_participates\_in}(1,0,0). \\
&\text{tr\_participates\_in}(1,0,1). \\
&\text{tr\_participates\_in}(1,1,0). \\
&\text{tr\_participates\_in}(1,1,1). \\
\end{align*}\]

\[H\_gangemi\_participation \models H\_PSL\_participates\]

**Proof attempt for axiom 1 of PSL\_participates:**

%common hierarchy
% Reading from file /stl/chui/macleod-master/qs/test/conversions/gangemi.p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (exists y g_pre(x,y))))).
(all x all y (g_pre(x,y) -> (g_Object(x) | g_Event(x)) & g_Time(y))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%psl_2_Tr
%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'
%
% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

%root theory:%Root Theory (collection of candidates’ signature term tautologies)

% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl Prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
%    dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
%    dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
%    dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%ax1
-((all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x)))
   )).

  Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.
f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0,0) = 0.
f5(0,1) = 0.
f5(1,0) = 0.
f5(1,1) = 0.

g_Event(0).
- g_Event(1).

g_Object(0).
- g_Object(1).

g_Time(0).
g_Time(1).

psl_activity_occurrence(0).
- psl_activity_occurrence(1).

- psl_object(0).
- psl_object(1).

psl_timepoint(0).
- psl_timepoint(1).

tr_event(0).
- tr_event(1).

tr_object(0).
- tr_object(1).

g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).

g_pre(0,0).
g_pre(0,1).
- g_pre(1,0).
Appendix B. Comparing Participation Ontologies: Proof Files

\[ H_{\text{gangemi}} \text{participation} \models H \cdot T_R \]

Proof attempt for CQ1 of \( T_R \):
%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/gangemi.p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (exists y g_pre(x,y))))).
(all x all y (g_pre(x,y) -> (g_Object(x) | g_Event(x)) & g_Time(y))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory:%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).
%psl_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).
(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ1 'Anything that participates in some event has some role when they are
  participating.'
-(all x all a all t
  (tr_participates_in(x,a,t))
& \text{tr\_event}(a)
\rightarrow
(exists z
\text{tr\_has\_role}(x,z,t))).$

**Counterexample found:**

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.
c2 = 0.
c3 = 0.

\text{psl\_beginof}(0) = 0.
\text{psl\_beginof}(1) = 0.

\text{psl\_endof}(0) = 0.
\text{psl\_endof}(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.

f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0,0) = 0.
f5(0,1) = 0.
f5(1,0) = 0.
f5(1,1) = 0.

g\_Event(0).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- g_Event(1).
  g_Object(0).
- g_Object(1).
  g_Time(0).
  g_Time(1).

tr_event(0).
- tr_event(1).

tr_object(0).
- tr_object(1).

  g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).

  g_pre(0,0).
  g_pre(0,1).
- g_pre(1,0).
- g_pre(1,1).

  tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).

  tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- \text{tr_participates\_in}(1, 0, 1).
- \text{tr_participates\_in}(1, 1, 0).
- \text{tr_participates\_in}(1, 1, 1).

\textbf{B.2.4} \quad H_{TR} \models T_j$

\text{H}_{TR} \models \text{H}_{PSL}.participates$

\textbf{Proof attempt for axiom 1 of PSL\_participates:}

\begin{verbatim}
% katsumi-thesis-searchcase-semantic-reqts-v1.clif
%minor correction to CQ4 (no impact on results, only changes signature)
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
 (tr_participates\_in(x, a, t)
  & tr\_event(a)
  ->
   (exists z
    tr\_has\_role(x, z, t))))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
 (tr_participates\_in(x, a, t)
  & tr\_event(a)
  ->
   tr\_exists\_at(x, t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
 (tr_participates\_in(x, a_1, t)
  & tr\_part\_of(a_1, a_2)
  ->
   tr_participates\_in(x, a_2, t))).

%CQ4 'Objects can not participate in more than one different activity.'
(all x all a_1 all a_2
 (tr\_object(x)
  & tr_participates\_in(x, a_1, t)
  & tr_participates\_in(x, a_2, t)
  ->
   ((a_1 = a_2) | tr\_part\_of(a_1, a_2)))).
\end{verbatim}
%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t (tr_participates_in(x,a,t) -> 
(tr_active_in(x,a,t) | tr_passive_in(x,a,t) | 
tr_consumed_in(x,a,t))))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) 
))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t))))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t))))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_person(x) -> tr_person(x))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%psl_2_Tr
%relevance mappings to requirements

%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).
% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of
   (o_1,o_2))).

%goal
%ax1:
-((all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))
     )).

   Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0,0,0) = 0.
f1(0,0,1) = 0.
f1(0,1,0) = 0.
f1(0,1,1) = 0.
f1(1,0,0) = 0.
f1(1,0,1) = 0.
f1(1,1,0) = 0.
f1(1,1,1) = 0.

   psl_activity_occurrence(0).
   - psl_activity_occurrence(1).

   - psl_object(0).
   - psl_object(1).

   psl_timepoint(0).
   - psl_timepoint(1).
tr_event(0).
- tr_event(1).
- tr_object(0).
- tr_object(1).
- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).
- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).

- tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
- tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).

\( H_{Tr} \models H_{gangemi} \)

Proof attempt for CQ1 of gangemi_participation:

\% katsumi-thesis-searchcase-semantic-reqts-v1.clif
%minor correction to CQ4 (no impact on results, only changes signature)

%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
 (tr_participates_in(x,a,t)
  & tr_event(a)
  ->
    (exists z
     tr_has_role(x,z,t))))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
 (tr_participates_in(x,a,t)
  & tr_event(a)
  ->
    tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
 (tr_participates_in(x,a_1,t)
  & tr_part_of(a_1,a_2)
  ->
    tr_participates_in(x,a_2,t))).

%CQ4 'Objects can not participate in more than one different activity.'
(all x all a_1 all a_2
 (tr_object(x)
  & tr_participates_in(x,a_1,t)
  & tr_participates_in(x,a_2,t)
  ->
    ((a_1 = a_2) | tr_part_of(a_1,a_2))))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
 (tr_participates_in(x,a,t)
  ->
    (tr_active_in(x,a,t) | tr_passive_in(x,a,t) | tr_consumed_in(x,a,t))))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_person(x) -> tr_person(x))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/ dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%gangemi_2_Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t)))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%goal
%ax1
-((all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))))

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

 t = 0.

c1 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f2(0,0) = 0.
f2(0,1) = 0.
f2(1,0) = 0.
f2(1,1) = 0.

f1(0,0,0) = 0.
f1(0,0,1) = 0.
f1(0,1,0) = 0.
f1(0,1,1) = 0.
f1(1,0,0) = 0.
f1(1,0,1) = 0.
f1(1,1,0) = 0.
f1(1,1,1) = 0.

g_Event(0).
g_Event(1).

-g_Object(0).
g_Object(1).

tr_event(0).
tr_event(1).

tr_object(0).
tr_object(1).

-g_hasParticipant(0,0).
g_hasParticipant(0,1).
g_hasParticipant(0,0).
g_hasParticipant(1,0).
g_hasParticipant(1,1).

-g_pre(0,0).
g_pre(0,1).
g_pre(1,0).
g_pre(1,1).

-tr_exists_at(0,0).
-tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
- tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).

$H_{TR} \models H_{DOLCE\_participation}$

**Proof attempt for axiom 1 of DOLCE_participation:**

%common hierarchy
% katsumi-thesis-searchcase-semantic-reqts-v1.clif
%minor correction to CQ4 (no impact on results, only changes signature)
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
 (tr_participates_in(x,a,t)
  & tr_event(a)
  ->
    (exists z
     tr_has_role(x,z,t))))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
 (tr_participates_in(x,a,t)
  & tr_event(a)
  ->
    tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
 (tr_participates_in(x,a_1,t)
  & tr_part_of(a_1,a_2)
  ->
    tr_participates_in(x,a_2,t))).

%CQ4 'Objects can not participate in more than one different activity.'
(all x all a_1 all a_2
(tr_object(x) & tr_participates_in(x,a_1,t) & tr_participates_in(x,a_2,t) -> ((a_1 = a_2) | tr_part_of(a_1,a_2))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t (tr_participates_in(x,a,t) -> (tr_active_in(x,a,t) | tr_passive_in(x,a,t) | tr_consumed_in(x,a,t)))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_person(x) -> tr_person(x))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[(\forall o \forall t \ (\text{psl_is_occuring_at}(o,t) \rightarrow \text{psl_is_occuring_at}(o,t))).\]
\[(\forall x \forall t \ (\text{psl_exists_at}(x,t) \rightarrow \text{psl_exists_at}(x,t))).\]
\[(\forall x \forall occ \forall t \ (\text{psl_participates_in}(x,occ,t) \rightarrow \text{psl_participates_in}(x,occ,t))).\]
\[(\forall t1 \forall t2 \forall t3 \ (\text{psl_between}(t1,t2,t3) \rightarrow \text{psl_between}(t1,t2,t3))).\]
\[(\forall t1 \forall t2 \ (\text{psl_before}(t1,t2) \rightarrow \text{psl_before}(t1,t2))).\]
\[(\forall t1 \forall t2 \ (\text{psl_beforeEq}(t1,t2) \rightarrow \text{psl_beforeEq}(t1,t2))).\]
\[(\forall t1 \forall t2 \forall t3 \ (\text{psl_betweenEq}(t1,t2,t3) \rightarrow \text{psl_betweenEq}(t1,t2,t3))).\]
\[\forall x \ \text{psl_endof}(x) = \text{psl_endof}(x).\]
\[\forall x \ \text{psl_beginof}(x) = \text{psl_beginof}(x).\]

%dolce_participation.p9 signature

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
\[(\forall x \forall y \forall t \ (\text{PC}(x,y,t) \rightarrow \text{PC}(x,y,t))).\]
\[(\forall x \ \text{ED}(x) \rightarrow \text{ED}(x))).\]
\[(\forall y \ \text{PD}(y) \rightarrow \text{PD}(y))).\]
\[\forall t \ \text{T}(t) \rightarrow \text{T}(t))).\]
\[\forall x \forall t \ (\text{PRE}(x,t) \rightarrow \text{PRE}(x,t))).\]
\[\forall t1 \forall t \ (\text{P}(t1,t) \rightarrow \text{P}(t1,t))).\]

%dolce_present.p9

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_present.p9
\[(\forall x \ (\text{Q}(x) \rightarrow \text{Q}(x))).\]
\[\forall t \forall t1 \forall t2 \ (\text{SUM}(t,t1,t2) \rightarrow \text{SUM}(t,t1,t2))).\]

%dolce_time_mereology.p9

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_time_mereology.p9
\[(\forall z \forall y \ (\text{O}(z,y) \rightarrow \text{O}(z,y))).\]
\[\forall z \forall y \ (\text{DJ}(z,y) \rightarrow \text{DJ}(z,y))).\]
\[\forall x \forall y \ (\text{PP}(x,y) \rightarrow \text{PP}(x,y))).\]
\[\forall x \forall y \ (\text{U}(x,y) \rightarrow \text{U}(x,y))).\]
\[\forall x \ (\text{AtP}(x) \rightarrow \text{AtP}(x))).\]
\[\forall x \forall y \ (\text{U}(x,y) \rightarrow \text{U}(x,y))).\]

%dolce_taxonomy.p9

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_taxonomy.p9
\[(\forall x \ (\text{AB}(x) \rightarrow \text{AB}(x))).\]
\[(\forall x \ (\text{PED}(x) \rightarrow \text{PED}(x))).\]
\[(\forall x \ (\text{NPED}(x) \rightarrow \text{NPED}(x))).\]
\[(\forall x \ (\text{AS}(x) \rightarrow \text{AS}(x))).\]


(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x)));

% dolce_2_Tr
% katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs

% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t)));

% 'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a)));

% 'no dolce_participation equivalent for tr_has_role'
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

%’PRE equivalent to tr_exists_at’
(all a all t   (PRE(a,t) <-> tr_exists_at(a,t))).

%’no dolce_participation equivalent for tr_part_of’

%’no dolce_participation equivalent for tr_object but we are given the
   additional info that an object is an endurant’)\%
   (all a (tr_object(a) -> ED(a))).

%’no dolce_participation equivalent for tr_active_in’

%’no dolce_participation equivalent for tr_passive_in’

%’no dolce_participation equivalent for tr_consumed_in’

%goal
%ax1
-((all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.
c2 = 0.
c3 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0,0,0) = 0.
f1(0,0,1) = 0.
f1(0,1,0) = 0.
\[ f_1(0,1,1) = 0. \]
\[ f_1(1,0,0) = 0. \]
\[ f_1(1,0,1) = 0. \]
\[ f_1(1,1,0) = 0. \]
\[ f_1(1,1,1) = 0. \]

- ED(0).
- ED(1).

- PD(0).
- PD(1).

- T(0).
- T(1).

- tr\_event(0).
- tr\_event(1).

- tr\_object(0).
- tr\_object(1).

- PRE(0,0).
- PRE(0,1).
- PRE(1,0).
- PRE(1,1).

- tr\_exists\_at(0,0).
- tr\_exists\_at(0,1).
- tr\_exists\_at(1,0).
- tr\_exists\_at(1,1).

- tr\_part\_of(0,0).
- tr\_part\_of(0,1).
- tr\_part\_of(1,0).
- tr\_part\_of(1,1).

PC(0,0,0).
- PC(0,0,1).
- PC(0,1,0).
- PC(0,1,1).
- PC(1,0,0).
- PC(1,0,1).
- PC(1,1,0).
- PC(1,1,1).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).

- tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
- tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(1,0,0).
B.3 Criteria 3 Proofs: \( T_i \cup T_R \models T_j \)

The purpose of these proofs is to assess whether the criteria: \( T_i \cup T_R \models T_j \) and \( \neg T_R \cup T_j \models T_i \) holds for any pair of candidates \( T_i, T_j \). We begin by attempting the first part of the criteria: \( T_i \cup T_R \models T_j \). As in the previous section, because the mappings weren’t syntactically applied to each theory, we include the mapping axioms for any theories involved in the antecedent of a given entailment problem to attain the shared signature. Proof or model output included where applicable.

B.3.1 \( T_i = PSL\_participates, T_j = gangemi\_participation \)

\( PSL\_participates \cup T_R \cup \Delta_{gangemi\rightarrow T_R} \models gangemi\_participation \)

Proof attempt for axiom 1 of gangemi_participation:

```prolog
%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
% psl_participates.p9  
% psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).  
(all x (psl_object(x) -> -psl_timepoint(x))).  
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) &  
  psl_timepoint(t))).  
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).  
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) &  
  psl_activity_occurrence(occ) & psl_timepoint(t))).  
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) &  
  psl_is_occurring_at(occ,t))).  
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) &  
  psl_before(t2,t3))).  
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(  
  t2) & (psl_before(t1,t2) | t1 = t2))).  
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) &  
  psl_beforeEq(t2,t3))).  
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(  
  x),t,psl_endof(x)))).  
(all occ all t (psl_is_occuring_at(occ,t) <-> psl_activity_occurrence(occ)  
  & psl_betweenEq(begnof(occ),t,endoof(occ)))).

%relevance mappings to requirements
```
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t    (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a    (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a    (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2    (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x    (g_Event(x) <-> tr_event(x))).
(all x    (g_Object(x) <-> tr_object(x))).
(all x all y    (g_hasParticipant(x,y) <-> (exists t    (tr_participates_in(y,x,t))))).
(all x all y    (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory
%Root Theory (collection of candidates' signature term tautologies)
%katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t    (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a    (tr_event(a) -> tr_event(a))).
(all x all z all t    (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t    (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2    (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x    (tr_object(x) -> tr_object(x))).
(all x all a all t    (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t    (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t    (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).
%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x)));
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y)));
(all x all y (g_pre(x,y) -> g_pre(x,y)));
(all y (g_Time(y) -> g_Time(y)));

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x)));
(all x (psl_object(x) -> psl_object(x)));
(all x (psl_timepoint(x) -> psl_timepoint(x)));
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t)));
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t)));
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t)));
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3)));
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2)));
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2)));
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1)));
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)));
(all x psl_endof(x) = psl_endof(x));
(all x psl_beginof(x) = psl_beginof(x));

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t)));
(all x (ED(x) -> ED(x)));
(all y (PD(y) -> PD(y)));
(all t (T(t) -> T(t)));
(all x all t (PRE(x,t) -> PRE(x,t)));
(all t1 all t (P(t1,t,t) -> P(t1,t,t)));

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x)));
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2)));
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).
(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%Tr
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
  (tr_participates_in(x,a,t) & tr_event(a)
  ->
    (exists z
      tr_has_role(x,z,t)))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
  (tr_participates_in(x,a,t) & tr_event(a)
  ->
    tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
  (tr_participates_in(x,a_1,t) & tr_part_of(a_1,a_2)
  ->
    tr_participates_in(x,a_2,t))).

%CQ4 'People can participate in more than one activity, but objects can not.'
(all x all a_1 all a_2
  (tr_object(x) & tr_participates_in(x,a_1,t)
  ->
    ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
  (tr_participates_in(x,a,t)
  ->
    ...
(tr_active_in(x,a,t) | tr_passive_in(x,a,t) | 
tr_consumed_in(x,a,t)))

%goal
%ax1

-((all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y)))))

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.

beginof(0) = 0.
beginof(1) = 0.

endof(0) = 0.
endof(1) = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0,0) = 0.
f1(0,1) = 0.
f1(1,0) = 0.
f1(1,1) = 0.

f2(0,0,0) = 0.
f2(0,0,1) = 0.
f2(0,1,0) = 0.
f2(0,1,1) = 0.
f2(1,0,0) = 0.
f2(1,0,1) = 0.
f2(1,1,0) = 0.
f2(1,1,1) = 0.
g_Event(0).
- g_Event(1).

- g_Object(0).
- g_Object(1).

- psl_activity_occurrence(0).
- psl_activity_occurrence(1).

- psl_object(0).
- psl_object(1).

- psl_timepoint(0).
- psl_timepoint(1).

- tr_event(0).
- tr_event(1).

- tr_object(0).
- tr_object(1).

- g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).

- g_pre(0,0).
- g_pre(0,1).
- g_pre(1,0).
- g_pre(1,1).

- psl_before(0,0).
- psl_before(0,1).
- psl_before(1,0).
- psl_before(1,1).

- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(1,0).
- psl_exists_at(1,1).

- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).

- psl_between(0,0,0).
- psl_between(0,0,1).
- psl_between(0,1,0).
- psl_between(0,1,1).
- psl_between(1,0,0).
- psl_between(1,0,1).
- psl_between(1,1,0).
- psl_between(1,1,1).

- psl_betweenEq(0,0,0).
- psl_betweenEq(0,0,1).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
B.3.2 \( T_i = \text{gangemi\_participation}, T_j = \text{PSL\_participates} \)

\[ \text{gangemi\_participation} \cup T_R \cup \Delta PSL \rightarrow T_R \models \text{PSL\_participates} \]

Proof attempt for axiom 1 of \text{PSL\_participates}:

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/gangemi.p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (exists y g_pre(x,y))))).
(all x all y (g_pre(x,y) -> (g_Object(x) | g_Event(x)) & g_Time(y))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%psl_2_Tr
%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t  (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a  (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a  (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2  (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

%root theory:%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t))))).
(all a  (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t))))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t))))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) → EV(x))).
(all x (STV(x) → STV(x))).
(all x (TQ(x) → TQ(x))).
(all x (PQ(x) → PQ(x))).
(all x (AQ(x) → AQ(x))).
(all x (R(x) → R(x))).
(all x (M(x) → M(x))).
(all x (F(x) → F(x))).
(all x (POB(x) → POB(x))).
(all x (NPOB(x) → NPOB(x))).
(all x (ACH(x) → ACH(x))).
(all x (ACC(x) → ACC(x))).
(all x (ST(x) → ST(x))).
(all x (PRO(x) → PRO(x))).
(all x (TL(x) → TL(x))).
(all x (SL(x) → SL(x))).
(all x (TR(x) → TR(x))).
(all x (PR(x) → PR(x))).
(all x (AR(x) → AR(x))).
(all x (APO(x) → APO(x))).
(all x (NAPO(x) → NAPO(x))).
(all x (MOB(x) → MOB(x))).
(all x (SOB(x) → SOB(x))).

(all x (S(x) → S(x))).
(all x (ASO(x) → ASO(x))).
(all x (NASO(x) → NASO(x))).
(all x (SAG(x) → SAG(x))).
(all x (SC(x) → SC(x))).
(all x (PT(x) → PT(x))).
(all x (PD(x) → PD(x))).

%Tr
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t (tr_participates_in(x,a,t) & tr_event(a) -> (exists z (tr_has_role(x,z,t))))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
   tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
  (tr_participates_in(x,a_1,t)
   & tr_part_of(a_1,a_2)
   ->
   tr_participates_in(x,a_2,t))).

%CQ4 'People can participate in more than one activity, but objects can not.'
(all x all a_1 all a_2
  (tr_object(x)
   & tr_participates_in(x,a_1,t)
   ->
   ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
  (tr_participates_in(x,a,t)
   ->
   (tr_active_in(x,a,t) | tr_passive_in(x,a,t) | tr_consumed_in(x,a,t)))).

%goal
%ax1
-((all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x)))).

  Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.

f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0,0) = 0.
f5(0,1) = 0.
f5(1,0) = 0.
f5(1,1) = 0.

f6(0,0,0) = 0.
f6(0,0,1) = 0.
f6(0,1,0) = 0.
f6(0,1,1) = 0.
f6(1,0,0) = 0.
f6(1,0,1) = 0.
f6(1,1,0) = 0.
f6(1,1,1) = 0.

g_Event(0).
- g_Event(1).

g_Object(0).
- g_Object(1).

g_Time(0).
g_Time(1).
psl_activity_occurrence(0).
- psl_activity_occurrence(1).

- psl_object(0).
- psl_object(1).

- psl_timepoint(0).
- psl_timepoint(1).

tr_event(0).
- tr_event(1).

tr_object(0).
- tr_object(1).

g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).

g_pre(0,0).
- g_pre(0,1).
- g_pre(1,0).
- g_pre(1,1).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(1,0).
- psl_exists_at(1,1).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).

  psl_participates_in(0,0,0).
  psl_participates_in(0,0,1).
  psl_participates_in(0,1,0).
  psl_participates_in(0,1,1).
  psl_participates_in(1,0,0).
  psl_participates_in(1,0,1).
  psl_participates_in(1,1,0).
  psl_participates_in(1,1,1).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).

  tr_has_role(0,0,0).
  tr_has_role(0,0,1).
  tr_has_role(0,1,0).
  tr_has_role(0,1,1).
  tr_has_role(1,0,0).
  tr_has_role(1,0,1).
  tr_has_role(1,1,0).
  tr_has_role(1,1,1).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
  tr_participates_in(0,1,0).
  tr_participates_in(0,1,1).
  tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).

  tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
  tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).

B.3.3 \( T_i = DOLCE_{participation}, T_j = PSL_{participates} \)

\( DOLCE_{participation} \cup T_R \models PSL_{participates} \)

Proof attempt for axiom 1 of PSL_{participates}:

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t))).
(all x (ED(x) -> (exists y exists t PC(x,y,t))).
(all x all y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t))).
(all x all y all t (PC(x,y,t) <-> (ED(x) & PD(y) & T(t)) & (all t1 ((P(t1,t)
  & T(t1)) -> PC(x,y,t1)))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t,t1,t2) -> PRE(x,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y))).
(all x all y (T(x) -> P(x,x))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & ~P(x,y) -> (exists z (T(z) & P(z,x) & ~O(z,y))))).
(all x all y (T(x) & T(y) & ~P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z))))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & ~P(y,x))))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z))))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> ~O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z))))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).
(all x (EV(x) | STV(x) -> PD(x))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).
(all x (R(x) -> AB(x))).
(all x (M(x) | F(x) | POB(x) -> PED(x))).
(all x (NPOB(x) -> NPED(x))).
(all x (ACH(x) | ACC(x) -> EV(x))).
(all x (ST(x) | PRO(x) -> STV(x))).
(all x (TL(x) -> TQ(x))).
(all x (SL(x) -> PQ(x))).
(all x (TR(x) | PR(x) | AR(x) -> R(x))).
(all x (APO(x) | NAPO(x) -> POB(x))).
(all x (MOB(x) | SOB(x) -> NPOB(x))).
(all x (T(x) -> TR(x))).
(all x (S(x) -> PR(x))).
(all x (ASO(x) | NASO(x) -> SOB(x))).
(all x (SAG(x) | SC(x) -> ASO(x))).
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).
(all x (ED(x) -> ~PD(x) & ~Q(x) & ~AB(x))).
(all x (PD(x) -> ~Q(x) & ~AB(x))).
(all x (Q(x) -> ~AB(x))).
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x)))).
(all x (PED(x) -> -NPED(x) & -AS(x)))).
(all x (NPED(x) -> -AS(x)))).
(all x (PD(x) <-> EV(x) | STV(x)))).
(all x (EV(x) -> -STV(x)))).
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x)))).
(all x (TQ(x) -> -PQ(x) & -AQ(x)))).
(all x (PQ(x) -> -AQ(x)))).
(all x (PED(x) <-> M(x) | F(x) | POB(x)))).
(all x (M(x) -> -F(x) & -POB(x)))).
(all x (F(x) -> -POB(x)))).
(all x (EV(x) <-> ACH(x) | ACC(x)))).
(all x (ACH(x) -> -ACC(x)))).
(all x (STV(x) <-> ST(x) | PRO(x)))).
(all x (ST(x) -> -PRO(x)))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x)))).
(all x (TR(x) -> -PR(x) & -AR(x)))).
(all x (PR(x) -> -AR(x)))).
(all x (POB(x) <-> APO(x) | NAPO(x)))).
(all x (APO(x) -> -NAPO(x)))).
(all x (NPOB(x) <-> MOB(x) | SOB(x)))).
(all x (MOB(x) -> -SOB(x)))).
(all x (SOB(x) <-> ASO(x) | NASO(x)))).
(all x (ASO(x) -> -NASO(x)))).
(all x (ASO(x) <-> SAG(x) | SC(x)))).
(all x (SAG(x) -> -SC(x)))).

%******relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t)))).

%‘PD equivalent to tr_event’
(all a (PD(a) <-> tr_event(a)))).

%‘no dolce_participation equivalent for tr_has_role’

%‘PRE equivalent to tr_exists_at’
(all a all t (PRE(a,t) <-> tr_exists_at(a,t)))).

%‘no dolce_participation equivalent for tr_part_of’
\% 'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant')
\%(all a (tr_object(a) -> ED(a))).

\% 'no dolce_participation equivalent for tr_active_in'

\% 'no dolce_participation equivalent for tr_passive_in'

\% 'no dolce_participation equivalent for tr_consumed_in'

\%psl_2_Tr
%relevance mappings to requirements

\%psl_2_Tr
%psl_prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(
  x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(
  o_1,o_2))).

********* root theory
%Root Theory (collection of candidates' signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)
  ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (FQ(x) -> FQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

% katsumi-thesis-searchcase-semantic-reqts-v1.clif

%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
 (tr_participates_in(x,a,t)
  & tr_event(a)
  ->
   (exists z
    tr_has_role(x,z,t)))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
 (tr_participates_in(x,a,t)
  & tr_event(a)
  ->
   tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
 (tr_participates_in(x,a_1,t)
  & tr_part_of(a_1,a_2)
  ->
   tr_participates_in(x,a_2,t))).

%CQ4 'People can participate in more than one activity, but objects can not.'
(all x all a_1 all a_2
 (tr_object(x)
  & tr_participates_in(x,a_1,t)
  ->
   ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
 (tr_participates_in(x,a,t)
  ->
(tr_active_in(x,a,t) | tr_passive_in(x,a,t) |
   tr_consumed_in(x,a,t))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) )).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).
%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PR(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%ax1:
-((all x (psl_activity_occurrence(x) -> ~psl_object(x) & ~psl_timepoint(x))
  )).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

t = 0.

c1 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.
psl_endof(2) = 0.
\[ f_2(0) = 0. \]
\[ f_2(1) = 0. \]
\[ f_2(2) = 0. \]

\[ f_3(0) = 0. \]
\[ f_3(1) = 0. \]
\[ f_3(2) = 1. \]

\[ f_5(0) = 1. \]
\[ f_5(1) = 0. \]
\[ f_5(2) = 1. \]

\[ f_{10}(0) = 0. \]
\[ f_{10}(1) = 0. \]
\[ f_{10}(2) = 0. \]

\[ f_1(0,0) = 0. \]
\[ f_1(0,1) = 2. \]
\[ f_1(0,2) = 0. \]
\[ f_1(1,0) = 0. \]
\[ f_1(1,1) = 0. \]
\[ f_1(1,2) = 0. \]
\[ f_1(2,0) = 0. \]
\[ f_1(2,1) = 0. \]
\[ f_1(2,2) = 0. \]

\[ f_6(0,0) = 0. \]
\[ f_6(0,1) = 0. \]
\[ f_6(0,2) = 0. \]
\[ f_6(1,0) = 0. \]
\[ f_6(1,1) = 0. \]
\[ f_6(1,2) = 0. \]
\[ f_6(2,0) = 0. \]
\[ f_6(2,1) = 0. \]
\[ f_6(2,2) = 0. \]

\[ f_7(0,0) = 0. \]
\[ f_7(0,1) = 0. \]
\[ f_7(0,2) = 0. \]
\[ f_7(1,0) = 0. \]
\[ f_7(1,1) = 0. \]
\[ f_7(1,2) = 0. \]
\[ f_7(2,0) = 0. \]
\[ f_7(2,1) = 0. \]
\[ f_7(2,2) = 0. \]

\[ f_8(0,0) = 0. \]
\[ f_8(0,1) = 0. \]
\[ f_8(0,2) = 0. \]
\[ f_8(1,0) = 0. \]
\[ f_8(1,1) = 1. \]
\[ f_8(1,2) = 0. \]
\[ f_8(2,0) = 0. \]
\[ f_8(2,1) = 0. \]
\[ f_8(2,2) = 0. \]

\[ f_9(0,0) = 0. \]
\[ f_9(0,1) = 0. \]
\[ f_9(0,2) = 0. \]
\[ f_9(1,0) = 0. \]
\[ f_9(1,1) = 1. \]
\[ f_9(1,2) = 0. \]
\[ f_9(2,0) = 0. \]
\[ f_9(2,1) = 0. \]
\[ f_9(2,2) = 0. \]

\[ f_{11}(0,0) = 0. \]
\[ f_{11}(0,1) = 0. \]
\[ f_{11}(0,2) = 0. \]
\[ f_{11}(1,0) = 0. \]
\[ f_{11}(1,1) = 1. \]
\[ f_{11}(1,2) = 0. \]
\[ f_{11}(2,0) = 0. \]
\[ f_{11}(2,1) = 0. \]
\[ f_{11}(2,2) = 0. \]

\[ f_{12}(0,0) = 0. \]
\[ f_{12}(0,1) = 0. \]
\[ f_{12}(0,2) = 0. \]
\[ f_{12}(1,0) = 0. \]
\[ f_{12}(1,1) = 1. \]
\[ f_{12}(1,2) = 0. \]
\[ f_{12}(2,0) = 0. \]
\[ f_{12}(2,1) = 0. \]
\[ f_{12}(2,2) = 0. \]
\[ f_{4}(0,0,0) = 0. \]
\[ f_{4}(0,0,1) = 0. \]
\[ f_{4}(0,0,2) = 0. \]
\[ f_{4}(0,1,0) = 0. \]
\[ f_{4}(0,1,1) = 0. \]
\[ f_{4}(0,1,2) = 0. \]
\[ f_{4}(0,2,0) = 0. \]
\[ f_{4}(0,2,1) = 0. \]
\[ f_{4}(0,2,2) = 0. \]
\[ f_{4}(1,0,0) = 0. \]
\[ f_{4}(1,0,1) = 0. \]
\[ f_{4}(1,0,2) = 0. \]
\[ f_{4}(1,1,0) = 0. \]
\[ f_{4}(1,1,1) = 0. \]
\[ f_{4}(1,1,2) = 0. \]
\[ f_{4}(1,2,0) = 0. \]
\[ f_{4}(1,2,1) = 0. \]
\[ f_{4}(1,2,2) = 0. \]
\[ f_{4}(2,0,0) = 0. \]
\[ f_{4}(2,0,1) = 0. \]
\[ f_{4}(2,0,2) = 0. \]
\[ f_{4}(2,1,0) = 0. \]
\[ f_{4}(2,1,1) = 0. \]
\[ f_{4}(2,1,2) = 0. \]
\[ f_{4}(2,2,0) = 0. \]
\[ f_{4}(2,2,1) = 0. \]
\[ f_{4}(2,2,2) = 0. \]

\[ f_{13}(0,0,0) = 0. \]
\[ f_{13}(0,0,1) = 0. \]
\[ f_{13}(0,0,2) = 0. \]
\[ f_{13}(0,1,0) = 0. \]
\[ f_{13}(0,1,1) = 0. \]
\[ f_{13}(0,1,2) = 0. \]
\[ f_{13}(0,2,0) = 0. \]
\[ f_{13}(0,2,1) = 0. \]
\[ f_{13}(0,2,2) = 0. \]
\[ f_{13}(1,0,0) = 0. \]
\[ f_{13}(1,0,1) = 0. \]
\[ f_{13}(1,0,2) = 0. \]
\[ f_{13}(1,1,0) = 0. \]
\[ f_{13}(1,1,1) = 0. \]
\[ f_{13}(1,1,2) = 0. \]
$f_{13}(1,2,0) = 0.$
$f_{13}(1,2,1) = 0.$
$f_{13}(1,2,2) = 0.$
$f_{13}(2,0,0) = 0.$
$f_{13}(2,0,1) = 0.$
$f_{13}(2,0,2) = 0.$
$f_{13}(2,1,0) = 0.$
$f_{13}(2,1,1) = 0.$
$f_{13}(2,1,2) = 0.$
$f_{13}(2,2,0) = 0.$
$f_{13}(2,2,1) = 0.$
$f_{13}(2,2,2) = 0.$

$f_{14}(0,0,0) = 0.$
$f_{14}(0,0,1) = 0.$
$f_{14}(0,0,2) = 0.$
$f_{14}(0,1,0) = 0.$
$f_{14}(0,1,1) = 0.$
$f_{14}(0,1,2) = 0.$
$f_{14}(0,2,0) = 0.$
$f_{14}(0,2,1) = 0.$
$f_{14}(0,2,2) = 0.$
$f_{14}(1,0,0) = 0.$
$f_{14}(1,0,1) = 0.$
$f_{14}(1,0,2) = 0.$
$f_{14}(1,1,0) = 0.$
$f_{14}(1,1,1) = 0.$
$f_{14}(1,1,2) = 0.$
$f_{14}(1,2,0) = 0.$
$f_{14}(1,2,1) = 0.$
$f_{14}(1,2,2) = 0.$
$f_{14}(2,0,0) = 0.$
$f_{14}(2,0,1) = 0.$
$f_{14}(2,0,2) = 0.$
$f_{14}(2,1,0) = 0.$
$f_{14}(2,1,1) = 0.$
$f_{14}(2,1,2) = 0.$
$f_{14}(2,2,0) = 0.$
$f_{14}(2,2,1) = 0.$
$f_{14}(2,2,2) = 0.$

- $AB(0).$
  
  $AB(1).$
- AB(2).
- ACC(0).
- ACC(1).
- ACC(2).
- ACH(0).
- ACH(1).
- ACH(2).
- APO(0).
- APO(1).
- APO(2).
- AQ(0).
- AQ(1).
- AQ(2).
- AR(0).
- AR(1).
- AR(2).
- AS(0).
- AS(1).
- AS(2).
- ASO(0).
- ASO(1).
- ASO(2).
- AtP(0).
- AtP(1).
- AtP(2).
- ED(0).
- ED(1).
- ED(2).
- EV(0).
- EV(1).
- EV(2).
- F(0).
- F(1).
- F(2).

- M(0).
- M(1).
- M(2).

- MOB(0).
- MOB(1).
- MOB(2).

- NAPO(0).
- NAPO(1).
- NAPO(2).

- NASO(0).
- NASO(1).
- NASO(2).

- NPED(0).
- NPED(1).
- NPED(2).

- NPOB(0).
- NPOB(1).
- NPOB(2).

- PD(0).
- PD(1).
- PD(2).

- PED(0).
- PED(1).
- PED(2).

- POB(0).
- POB(1).
- POB(2).

- PQ(0).
- PQ(1).
- PQ(2).
- PR(0).
- PR(1).
- PR(2).

- PRO(0).
- PRO(1).
- PRO(2).

- PT(0).
- PT(1).
- PT(2).

- Q(0).
- Q(1).
- Q(2).

- R(0).
- R(1).
- R(2).

- S(0).
- S(1).
- S(2).

- SAG(0).
- SAG(1).
- SAG(2).

- SC(0).
- SC(1).
- SC(2).

- SL(0).
- SL(1).
- SL(2).

- SOB(0).
- SOB(1).
- SOB(2).

- ST(0).
- ST(1).
- ST(2).
STV(0).
- STV(1).
- STV(2).

- T(0).
  T(1).
- T(2).

- TL(0).
- TL(1).
- TL(2).

- TQ(0).
- TQ(1).
- TQ(2).

- TR(0).
  TR(1).
- TR(2).

  psl_activity_occurrence(0).
- psl_activity_occurrence(1).
- psl_activity_occurrence(2).

  psl_object(0).
  psl_object(1).
- psl_object(2).

  psl_timepoint(0).
- psl_timepoint(1).
- psl_timepoint(2).

  tr_event(0).
- tr_event(1).
- tr_event(2).

  tr_object(0).
- tr_object(1).
- tr_object(2).

  DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).
- O(0,0).
- O(0,1).
- O(0,2).
- O(1,0).
- O(1,1).
- O(1,2).
- O(2,0).
- O(2,1).
- O(2,2).
- P(0,0).
- P(0,1).
- P(0,2).
- P(1,0).
- P(1,1).
- P(1,2).
- P(2,0).
- P(2,1).
- P(2,2).
- PP(0,0).
- PP(0,1).
- PP(0,2).
- PP(1,0).
- PP(1,1).
- PP(1,2).
- PP(2,0).
- PP(2,1).
- PP(2,2).
- PRE(0,0).
- PRE(0,1).
- PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- \text{PRE}(1,2).
- \text{PRE}(2,0).
- \text{PRE}(2,1).
- \text{PRE}(2,2).

- \text{U}(0,0).
- \text{U}(0,1).
- \text{U}(0,2).
- \text{U}(1,0).
- \text{U}(1,1).
- \text{U}(1,2).
- \text{U}(2,0).
- \text{U}(2,1).
- \text{U}(2,2).

- \text{psl}_\text{exists}_\text{at}(0,0).
- \text{psl}_\text{exists}_\text{at}(0,1).
- \text{psl}_\text{exists}_\text{at}(0,2).
- \text{psl}_\text{exists}_\text{at}(1,0).
- \text{psl}_\text{exists}_\text{at}(1,1).
- \text{psl}_\text{exists}_\text{at}(1,2).
- \text{psl}_\text{exists}_\text{at}(2,0).
- \text{psl}_\text{exists}_\text{at}(2,1).
- \text{psl}_\text{exists}_\text{at}(2,2).

- \text{psl}_\text{subactivity}_\text{occurrence}(0,0).
- \text{psl}_\text{subactivity}_\text{occurrence}(0,1).
- \text{psl}_\text{subactivity}_\text{occurrence}(0,2).
- \text{psl}_\text{subactivity}_\text{occurrence}(1,0).
- \text{psl}_\text{subactivity}_\text{occurrence}(1,1).
- \text{psl}_\text{subactivity}_\text{occurrence}(1,2).
- \text{psl}_\text{subactivity}_\text{occurrence}(2,0).
- \text{psl}_\text{subactivity}_\text{occurrence}(2,1).
- \text{psl}_\text{subactivity}_\text{occurrence}(2,2).

- \text{tr}_\text{exists}_\text{at}(0,0).
- \text{tr}_\text{exists}_\text{at}(0,1).
- \text{tr}_\text{exists}_\text{at}(0,2).
- \text{tr}_\text{exists}_\text{at}(1,0).
- \text{tr}_\text{exists}_\text{at}(1,1).
- \text{tr}_\text{exists}_\text{at}(1,2).
- \text{tr}_\text{exists}_\text{at}(2,0).
- \text{tr}_\text{exists}_\text{at}(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,0,2).
- PC(0,1,0).
- PC(0,1,1).
- PC(0,1,2).
- PC(0,2,0).
- PC(0,2,1).
- PC(0,2,2).
- PC(1,0,0).
- PC(1,0,1).
- PC(1,0,2).
- PC(1,1,0).
- PC(1,1,1).
- PC(1,1,2).
- PC(1,2,0).
- PC(1,2,1).
- PC(1,2,2).
- PC(2,0,0).
- PC(2,0,1).
- PC(2,0,2).
- PC(2,1,0).
- PC(2,1,1).
- PC(2,1,2).
- PC(2,2,0).
- PC(2,2,1).
- PC(2,2,2).

- SUM(0,0,0).
- SUM(0,0,1).
- SUM(0,0,2).
- SUM(0,1,0).
- SUM(0,1,1).
- SUM(0,1,2).
- SUM(0,2,0).
- SUM(0,2,1).
- SUM(0,2,2).
- SUM(1,0,0).
- SUM(1,0,1).
- SUM(1,0,2).
- SUM(1,1,0).
- SUM(1,1,1).
- SUM(1,1,2).
- SUM(1,2,0).
- SUM(1,2,1).
- SUM(1,2,2).
- SUM(2,0,0).
- SUM(2,0,1).
- SUM(2,0,2).
- SUM(2,1,0).
- SUM(2,1,1).
- SUM(2,1,2).
- SUM(2,2,0).
- SUM(2,2,1).
- SUM(2,2,2).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
  psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,0,2).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(0,1,2).
- tr_active_in(0,2,0).
- tr_active_in(0,2,1).
- tr_active_in(0,2,2).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,0,2).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).
- tr_active_in(1,1,2).
- tr_active_in(1,2,0).
- tr_active_in(1,2,1).
- tr_active_in(1,2,2).
- tr_active_in(2,0,0).
- tr_active_in(2,0,1).
- tr_active_in(2,0,2).
- tr_active_in(2,1,0).
- tr_active_in(2,1,1).
- tr_active_in(2,1,2).
- tr_active_in(2,2,0).
- tr_active_in(2,2,1).
- tr_active_in(2,2,2).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,0,2).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(0,1,2).
- tr_consumed_in(0,2,0).
- tr_consumed_in(0,2,1).
- tr_consumed_in(0,2,2).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,0,2).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).
- tr_consumed_in(1,1,2).
- tr_consumed_in(1,2,0).
- tr_consumed_in(1,2,1).
- tr_consumed_in(1,2,2).
- tr_consumed_in(2,0,0).
- tr_consumed_in(2,0,1).
- tr_consumed_in(2,0,2).
- tr_consumed_in(2,1,0).
- tr_consumed_in(2,1,1).
- tr_consumed_in(2,1,2).
- tr_consumed_in(2,2,0).
- tr_consumed_in(2,2,1).
- tr_consumed_in(2,2,2).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,0,2).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(0,1,2).
- tr_has_role(0,2,0).
- tr_has_role(0,2,1).
- tr_has_role(0,2,2).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,0,2).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).
- tr_has_role(1,1,2).
- tr_has_role(1,2,0).
- tr_has_role(1,2,1).
- tr_has_role(1,2,2).
- tr_has_role(2,0,0).
- tr_has_role(2,0,1).
- tr_has_role(2,0,2).
- tr_has_role(2,1,0).
- tr_has_role(2,1,1).
- tr_has_role(2,1,2).
- tr_has_role(2,2,0).
- tr_has_role(2,2,1).
- tr_has_role(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

- tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
- tr_passive_in(0,0,2).
- tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(0,1,2).
- tr_passive_in(0,2,0).
- tr_passive_in(0,2,1).
- tr_passive_in(0,2,2).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,0,2).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).
- tr_passive_in(1,1,2).
- tr_passive_in(1,2,0).
- tr_passive_in(1,2,1).
- tr_passive_in(1,2,2).
- tr_passive_in(2,0,0).
- tr_passive_in(2,0,1).
- tr_passive_in(2,0,2).
- tr_passive_in(2,1,0).
- tr_passive_in(2,1,1).
- tr_passive_in(2,1,2).
- tr_passive_in(2,2,0).
- tr_passive_in(2,2,1).
- tr_passive_in(2,2,2).

B.3.4 $T_i = PSL\_participates, T_j = DOLCE\_participation$

$PSL\_participates \cup T_R \cup T_j \models DOLCE\_participation$

Proof attempt for axiom 1 of DOLCE\_participation:

%common hierarchy
% Reading from file /stl/chui/macleod-master/qs/test/conversions/  
  psl_participates.p9
%psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).
(all x (psl_object(x) -> -psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) &  
  psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) &  
  psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) &  
  psl_is_occurring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) &  
  psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) & (psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) & psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,psl_endof(x)))).
(all occ all t (psl_is_occurring_at(occ,t) <-> psl_activity_occurrence(occ) & psl_betweenEq(beginof(occ),t,endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'no psl equivalent for tr_has_role'

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

%*****relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).
Appendix B. Comparing Participation Ontologies: Proof Files

%'no dolce_participation equivalent for tr_part_of'

%'no dolce_participation equivalent for tr_object but we are given the
  additional info that an object is an endurant’)
%(all a (tr_object(a) -> ED(a))).

%'no dolce_participation equivalent for tr_active_in'

%'no dolce_participation equivalent for tr_passive_in'

%'no dolce_participation equivalent for tr_consumed_in'

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)
   )))
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x
   ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%Tr
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
    (exists z
     tr_has_role(x,z,t)))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->

tr_exists_at(x,t)).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
  (tr_participates_in(x,a_1,t)
   & tr_part_of(a_1,a_2)
   ->
    tr_participates_in(x,a_2,t))).

%CQ4 'People can participate in more than one activity, but objects can not.
'(all x all a_1 all a_2
  (tr_object(x)
   & tr_participates_in(x,a_1,t)
   ->
    ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
  (tr_participates_in(x,a,t)
   ->
    (tr_active_in(x,a,t) | tr_passive_in(x,a,t) | tr_consumed_in(x,a,t)))).

%goal
%ax1
-((all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

t = 0.
c1 = 0.
c2 = 1.
c3 = 2.
Appendix B. Comparing Participation Ontologies: Proof Files

beginof(0) = 2.
beginof(1) = 2.
beginof(2) = 0.

endof(0) = 0.
endof(1) = 2.
endof(2) = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 2.
psl_endof(1) = 0.
psl_endof(2) = 0.

f1(0,0,0) = 0.
f1(0,0,1) = 0.
f1(0,0,2) = 0.
f1(0,1,0) = 0.
f1(0,1,1) = 0.
f1(0,1,2) = 0.
f1(0,2,0) = 0.
f1(0,2,1) = 0.
f1(0,2,2) = 0.
f1(1,0,0) = 0.
f1(1,0,1) = 0.
f1(1,0,2) = 0.
f1(1,1,0) = 0.
f1(1,1,1) = 0.
f1(1,1,2) = 0.
f1(1,2,0) = 0.
f1(1,2,1) = 0.
f1(1,2,2) = 0.
f1(2,0,0) = 0.
f1(2,0,1) = 0.
f1(2,0,2) = 0.
f1(2,1,0) = 0.
f1(2,1,1) = 0.
f1(2,1,2) = 0.
f1(2,2,0) = 0.
f1(2,2,1) = 0.
f1(2,2,2) = 0.

- ED(0).
- ED(1).
- ED(2).

- PD(0).
  PD(1).
- PD(2).

- T(0).
- T(1).
- T(2).

- psl_activity_occurrence(0).
  psl_activity_occurrence(1).
- psl_activity_occurrence(2).

  psl_object(0).
- psl_object(1).
  psl_object(2).

  psl_timepoint(0).
  psl_timepoint(1).
- psl_timepoint(2).

- tr_event(0).
  tr_event(1).
- tr_event(2).

- tr_object(0).
- tr_object(1).
- tr_object(2).

- PRE(0,0).
- PRE(0,1).
  PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- PRE(1,2).
- PRE(2,0).
- PRE(2,1).
- PRE(2,2).
- psl_before(0,0).
- psl_before(0,1).
- psl_before(0,2).
- psl_before(1,0).
- psl_before(1,1).
- psl_before(1,2).
- psl_before(2,0).
- psl_before(2,1).
- psl_before(2,2).
- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(0,2).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).
- psl_beforeEq(1,2).
- psl_beforeEq(2,0).
- psl_beforeEq(2,1).
- psl_beforeEq(2,2).
- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(0,2).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).
- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(0,2).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).
- psl_is_occuring_at(1,2).
- psl_is_occuring_at(2,0).
- psl_is_occuring_at(2,1).
- psl_is_occuring_at(2,2).
- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(2,0).
- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,0,2).
- PC(0,1,0).
- PC(0,1,1).
- PC(0,1,2).
- PC(0,2,0).
- PC(0,2,1).
- PC(0,2,2).
- PC(1,0,0).
- PC(1,0,1).
- PC(1,0,2).
- PC(1,1,0).
- PC(1,1,1).
- PC(1,1,2).
- \text{PC}(1,2,0).
- \text{PC}(1,2,1).
- \text{PC}(1,2,2).
- \text{PC}(2,0,0).
- \text{PC}(2,0,1).
- \text{PC}(2,0,2).
- \text{PC}(2,1,0).
- \text{PC}(2,1,1).
- \text{PC}(2,1,2).
- \text{PC}(2,2,0).
- \text{PC}(2,2,1).
- \text{PC}(2,2,2).

- \text{psl\_between}(0,0,0).
- \text{psl\_between}(0,0,1).
- \text{psl\_between}(0,0,2).
- \text{psl\_between}(0,1,0).
- \text{psl\_between}(0,1,1).
- \text{psl\_between}(0,1,2).
- \text{psl\_between}(0,2,0).
- \text{psl\_between}(0,2,1).
- \text{psl\_between}(0,2,2).
- \text{psl\_between}(1,0,0).
- \text{psl\_between}(1,0,1).
- \text{psl\_between}(1,0,2).
- \text{psl\_between}(1,1,0).
- \text{psl\_between}(1,1,1).
- \text{psl\_between}(1,1,2).
- \text{psl\_between}(1,2,0).
- \text{psl\_between}(1,2,1).
- \text{psl\_between}(1,2,2).
- \text{psl\_between}(2,0,0).
- \text{psl\_between}(2,0,1).
- \text{psl\_between}(2,0,2).
- \text{psl\_between}(2,1,0).
- \text{psl\_between}(2,1,1).
- \text{psl\_between}(2,1,2).
- \text{psl\_between}(2,2,0).
- \text{psl\_between}(2,2,1).
- \text{psl\_between}(2,2,2).

- \text{psl\_betweenEq}(0,0,0).
- \text{psl\_betweenEq}(0,0,1).
- psl_betweenEq(0,0,2).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(0,1,2).
- psl_betweenEq(0,2,0).
- psl_betweenEq(0,2,1).
- psl_betweenEq(0,2,2).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,0,2).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).
- psl_betweenEq(1,1,2).
- psl_betweenEq(1,2,0).
- psl_betweenEq(1,2,1).
- psl_betweenEq(1,2,2).
- psl_betweenEq(2,0,0).
- psl_betweenEq(2,0,1).
- psl_betweenEq(2,0,2).
- psl_betweenEq(2,1,0).
- psl_betweenEq(2,1,1).
- psl_betweenEq(2,1,2).
- psl_betweenEq(2,2,0).
- psl_betweenEq(2,2,1).
- psl_betweenEq(2,2,2).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,0,2).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(0,1,2).
- tr_active_in(0,2,0).
- tr_active_in(0,2,1).
- tr_active_in(0,2,2).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,0,2).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).
- tr_active_in(1,1,2).
- tr_active_in(1,2,0).
- tr_active_in(1,2,1).
- tr_active_in(1,2,2).
- tr_active_in(2,0,0).
- tr_active_in(2,0,1).
- tr_active_in(2,0,2).
- tr_active_in(2,1,0).
- tr_active_in(2,1,1).
- tr_active_in(2,1,2).
- tr_active_in(2,2,0).
- tr_active_in(2,2,1).
- tr_active_in(2,2,2).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,0,2).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(0,1,2).
- tr_consumed_in(0,2,0).
- tr_consumed_in(0,2,1).
- tr_consumed_in(0,2,2).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,0,2).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).
- tr_consumed_in(1,1,2).
- tr_consumed_in(1,2,0).
- tr_consumed_in(1,2,1).
- tr_consumed_in(1,2,2).
- tr_consumed_in(2,0,0).
- tr_consumed_in(2,0,1).
- tr_consumed_in(2,0,2).
- tr_consumed_in(2,1,0).
- tr_consumed_in(2,1,1).
- tr_consumed_in(2,1,2).
- tr_consumed_in(2,2,0).
- tr_consumed_in(2,2,1).
- tr_consumed_in(2,2,2).

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,0,2).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(0,1,2).
- tr_has_role(0,2,0).
- tr_has_role(0,2,1).
- tr_has_role(0,2,2).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,0,2).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).
- tr_has_role(1,1,2).
- tr_has_role(1,2,0).
- tr_has_role(1,2,1).
- tr_has_role(1,2,2).
- tr_has_role(2,0,0).
- tr_has_role(2,0,1).
- tr_has_role(2,0,2).
- tr_has_role(2,1,0).
- tr_has_role(2,1,1).
- tr_has_role(2,1,2).
- tr_has_role(2,2,0).
- tr_has_role(2,2,1).
- tr_has_role(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

- tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
- tr_passive_in(0,0,2).
- tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(0,1,2).
- tr_passive_in(0,2,0).
- tr_passive_in(0,2,1).
- tr_passive_in(0,2,2).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,0,2).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).
- tr_passive_in(1,1,2).
- tr_passive_in(1,2,0).
- tr_passive_in(1,2,1).
- tr_passive_in(1,2,2).
- tr_passive_in(2,0,0).
- tr_passive_in(2,0,1).
- tr_passive_in(2,0,2).
- tr_passive_in(2,1,0).
- tr_passive_in(2,1,1).
- tr_passive_in(2,1,2).
- tr_passive_in(2,2,0).
- tr_passive_in(2,2,1).
- tr_passive_in(2,2,2).

B.3.5 $T_i = \textit{DOLCE\_participation}, T_j = \textit{gangemi\_participation}$

$\textit{DOLCE\_participation} \cup T_R \models \textit{gangemi\_participation}$

Proof attempt for axiom 1 of $\textit{gangemi\_participation}$:

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t))).
(all x (ED(x) -> (exists y exists t PC(x,y,t))).
(all x y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t))).
(all x y all t (PC(x,y,t) <-> (ED(x) & PD(y) & T(t) & (all t1 ((P(t1,t)
 & T(t1)) -> PC(x,y,t1)))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t1,t2) -> PRE(x,t))).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y)))).
(all x all y (T(x) -> P(x,y))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (T(z) & P(z,x) & D(z,y) & T(z)))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z))))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & R(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y))))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z))))).
(all x all y (T(x) & T(y) -> (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).
(all x (EV(x) | STV(x) -> PD(x))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).
(all x (R(x) -> AB(x)))).
(all x (M(x) | F(x) | POB(x) -> PED(x))).
(all x (NPOB(x) -> NPED(x))).
(all x (ACH(x) | ACC(x) -> EV(x)))).
(all x (ST(x) | PRO(x) -> STV(x))).
(all x (TL(x) -> TQ(x)))).
(all x (SL(x) -> PQ(x)))).
(all x (TR(x) | PR(x) | AR(x) -> R(x)))).
(all x (APO(x) | NAPO(x) -> POB(x)))).
(all x (MOB(x) | SOB(x) -> NPOB(x)))).
(all x (T(x) -> TR(x))).
(all x (S(x) -> PR(x))).
(all x (ASO(x) | NASO(x) -> SOB(x))).
(all x (SAG(x) | SC(x) -> ASO(x))).
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).
(all x (ED(x) -> -PD(x) & -Q(x) & -AB(x))).
(all x (PD(x) -> -Q(x) & -AB(x))).
(all x (Q(x) -> -AB(x))).
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x))).
(all x (PED(x) -> -NPED(x) & -AS(x))).
(all x (NPED(x) -> -AS(x))).
(all x (PD(x) <-> EV(x) | STV(x))).
(all x (EV(x) -> -STV(x))).
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x))).
(all x (TQ(x) -> -PQ(x) & -AQ(x))).
(all x (PQ(x) -> -AQ(x))).
(all x (PED(x) <-> M(x) | F(x) | POB(x))).
(all x (M(x) -> -F(x) & -POB(x))).
(all x (F(x) -> -POB(x))).
(all x (EV(x) <-> ACH(x) | ACC(x))).
(all x (ACH(x) -> -ACC(x))).
(all x (STV(x) <-> ST(x) | PRO(x))).
(all x (ST(x) -> -PRO(x))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> -PR(x) & -AR(x))).
(all x (PR(x) -> -AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> -NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> -SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> -NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> -SC(x))).

%******relevance mappings to requirements
%dolce_2_tr
%katumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% ’PC equivalent to tr_participates_in’
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%’PD equivalent to tr_event’
(all a (PD(a) <-> tr_event(a))).
%‘no dolce_participation equivalent for tr_has_role’

%‘PRE equivalent to tr_exists_at’
(all a all t  (PRE(a,t) <-> tr_exists_at(a,t))).

%‘no dolce_participation equivalent for tr_part_of’

%‘no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant’) % (all a (tr_object(a) -> ED(a))).

%‘no dolce_participation equivalent for tr_active_in’

%‘no dolce_participation equivalent for tr_passive_in’

%‘no dolce_participation equivalent for tr_consumed_in’

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%**********root theory
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) )).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).
%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (Ev(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).
(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

% katsumi-thesis-searchcase-semantic-reqts-v1.clif

%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
   (tr_participates_in(x,a,t)
    & tr_event(a)
    ->
    (exists z
     tr_has_role(x,z,t)))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
   (tr_participates_in(x,a,t)
    & tr_event(a)
    ->
    tr_exists_at(x,t))).

%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
   (tr_participates_in(x,a_1,t)
    & tr_part_of(a_1,a_2)
    ->
    tr_participates_in(x,a_2,t))).

%CQ4 'People can participate in more than one activity, but objects can not.'
(all x all a_1 all a_2
   (tr_object(x)
    & tr_participates_in(x,a_1,t)
    ->
    ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
   (tr_participates_in(x,a,t)
    ->
\[(\text{tr\_active\_in}(x,a,t) \mid \text{tr\_passive\_in}(x,a,t) \mid \text{tr\_consumed\_in}(x,a,t))).\]

%root theory
%Root Theory (collection of candidates' signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (\text{tr\_participates\_in}(x,a,t) \rightarrow (\text{tr\_participates\_in}(x,a,t)))
).
(all a (\text{tr\_event}(a) \rightarrow \text{tr\_event}(a))
).
(all x all z all t (\text{tr\_has\_role}(x,z,t) \rightarrow (\text{tr\_has\_role}(x,z,t)))
).
(all x all t (\text{tr\_exists\_at}(x,t) \rightarrow (\text{tr\_exists\_at}(x,t)))
).
(all a_1 all a_2 (\text{tr\_part\_of}(a_1,a_2) \rightarrow \text{tr\_part\_of}(a_1,a_2))
).
(all x (\text{tr\_object}(x) \rightarrow \text{tr\_object}(x))
).
(all x all a all t (\text{tr\_active\_in}(x,a,t) \rightarrow \text{tr\_active\_in}(x,a,t))
).
(all x all a all t (\text{tr\_passive\_in}(x,a,t) \rightarrow \text{tr\_passive\_in}(x,a,t))
).
(all x all a all t (\text{tr\_consumed\_in}(x,a,t) \rightarrow \text{tr\_consumed\_in}(x,a,t))
).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g\_Event(x) \rightarrow g\_Event(x)))
).
(all x (g\_Object(x) \rightarrow g\_Object(x))
).
(all x all y (g\_hasParticipant(x,y) \rightarrow g\_hasParticipant(x,y))
).
(all x all y (g\_pre(x,y) \rightarrow g\_pre(x,y))
).
(all y (g\_Time(y) \rightarrow g\_Time(y))
).

%psl\_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl\_activity\_occurrence(x) \rightarrow psl\_activity\_occurrence(x)))
).
(all x (psl\_object(x) \rightarrow psl\_object(x))
).
(all x (psl\_timepoint(x) \rightarrow psl\_timepoint(x))
).
(all o all t (psl\_is\_occurring\_at(o,t) \rightarrow psl\_is\_occurring\_at(o,t))
).
(all x all t (psl\_exists\_at(x,t) \rightarrow psl\_exists\_at(x,t))
).
(all x all occ all t (psl\_participates\_in(x,occ,t) \rightarrow psl\_participates\_in(x,occ,t))
).
(all t1 all t2 all t3 (psl\_between(t1,t2,t3) \rightarrow psl\_between(t1,t2,t3))
).
(all t1 all t2 (psl\_before(t1,t2) \rightarrow psl\_before(t1,t2))
).
(all t1 all t2 (psl\_beforeEq(t1,t2) \rightarrow psl\_beforeEq(t1,t2))
).
(all t1 (psl\_timepoint(t1) \rightarrow psl\_timepoint(t1))
).
(all t1 all t2 all t3 (psl\_betweenEq(t1,t2,t3) \rightarrow psl\_betweenEq(t1,t2,t3))
).
(all x psl\_endof(x) = psl\_endof(x))
).
(all x psl\_beginof(x) = psl\_beginof(x)).
%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
\[(\text{all } x \ (\text{ST}(x) \to \text{ST}(x)))\].
\[(\text{all } x \ (\text{PRO}(x) \to \text{PRO}(x)))\].
\[(\text{all } x \ (\text{TL}(x) \to \text{TL}(x)))\].
\[(\text{all } x \ (\text{SL}(x) \to \text{SL}(x)))\].
\[(\text{all } x \ (\text{TR}(x) \to \text{TR}(x)))\].
\[(\text{all } x \ (\text{F}(x) \to \text{F}(x)))\].
\[(\text{all } x \ (\text{AR}(x) \to \text{AR}(x)))\].
\[(\text{all } x \ (\text{APO}(x) \to \text{APO}(x)))\].
\[(\text{all } x \ (\text{NAPO}(x) \to \text{NAPO}(x)))\].
\[(\text{all } x \ (\text{MOB}(x) \to \text{MOB}(x)))\].
\[(\text{all } x \ (\text{SOB}(x) \to \text{SOB}(x)))\].

\[(\text{all } x \ (\text{S}(x) \to \text{S}(x)))\].
\[(\text{all } x \ (\text{ASO}(x) \to \text{ASO}(x)))\].
\[(\text{all } x \ (\text{NASO}(x) \to \text{NASO}(x)))\].
\[(\text{all } x \ (\text{SAG}(x) \to \text{SAG}(x)))\].
\[(\text{all } x \ (\text{SC}(x) \to \text{SC}(x)))\].
\[(\text{all } x \ (\text{FT}(x) \to \text{FT}(x)))\].
\[(\text{all } x \ (\text{PD}(x) \to \text{PD}(x)))\].

\%goal
\[-((\text{all } x \ (\text{Event}(x) \to (\exists \ y \ (\text{Object}(x) \& \text{hasParticipant}(x,y))))))\].

\textbf{Counterexample found:}

\% number = 1
\% seconds = 0

\% Interpretation of size 3
\n\% t = 0.
\n\% c1 = 0.
\n\% psl\_beginof(0) = 0.
\% psl\_beginof(1) = 0.
\% psl\_beginof(2) = 0.
\n\% psl\_endof(0) = 0.
\% psl\_endof(1) = 0.
\% psl\_endof(2) = 0.
\n\% f2(0) = 0.
\begin{verbatim}
f2(1) = 0.
f2(2) = 0.

f3(0) = 0.
f3(1) = 0.
f3(2) = 1.

f5(0) = 1.
f5(1) = 0.
f5(2) = 1.

f10(0) = 0.
f10(1) = 0.
f10(2) = 0.

f1(0,0) = 0.
f1(0,1) = 2.
f1(0,2) = 0.
f1(1,0) = 0.
f1(1,1) = 0.
f1(1,2) = 0.
f1(2,0) = 0.
f1(2,1) = 0.
f1(2,2) = 0.

f6(0,0) = 0.
f6(0,1) = 0.
f6(0,2) = 0.
f6(1,0) = 0.
f6(1,1) = 0.
f6(1,2) = 0.
f6(2,0) = 0.
f6(2,1) = 0.
f6(2,2) = 0.

f7(0,0) = 0.
f7(0,1) = 0.
f7(0,2) = 0.
f7(1,0) = 0.
f7(1,1) = 0.
f7(1,2) = 0.
f7(2,0) = 0.
f7(2,1) = 0.
\end{verbatim}
f7(2,2) = 0.

f8(0,0) = 0.
f8(0,1) = 0.
f8(0,2) = 0.
f8(1,0) = 0.
f8(1,1) = 1.
f8(1,2) = 0.
f8(2,0) = 0.
f8(2,1) = 0.
f8(2,2) = 0.

f9(0,0) = 0.
f9(0,1) = 0.
f9(0,2) = 0.
f9(1,0) = 0.
f9(1,1) = 1.
f9(1,2) = 0.
f9(2,0) = 0.
f9(2,1) = 0.
f9(2,2) = 0.

f11(0,0) = 0.
f11(0,1) = 0.
f11(0,2) = 0.
f11(1,0) = 0.
f11(1,1) = 1.
f11(1,2) = 0.
f11(2,0) = 0.
f11(2,1) = 0.
f11(2,2) = 0.

f12(0,0) = 0.
f12(0,1) = 0.
f12(0,2) = 0.
f12(1,0) = 0.
f12(1,1) = 1.
f12(1,2) = 0.
f12(2,0) = 0.
f12(2,1) = 0.
f12(2,2) = 0.

f14(0,0) = 0.
\[ f_{14}(0,1) = 0. \]
\[ f_{14}(0,2) = 1. \]
\[ f_{14}(1,0) = 0. \]
\[ f_{14}(1,1) = 0. \]
\[ f_{14}(1,2) = 0. \]
\[ f_{14}(2,0) = 0. \]
\[ f_{14}(2,1) = 0. \]
\[ f_{14}(2,2) = 0. \]

\[ f_{4}(0,0,0) = 0. \]
\[ f_{4}(0,0,1) = 0. \]
\[ f_{4}(0,0,2) = 0. \]
\[ f_{4}(0,1,0) = 0. \]
\[ f_{4}(0,1,1) = 0. \]
\[ f_{4}(0,1,2) = 0. \]
\[ f_{4}(0,2,0) = 0. \]
\[ f_{4}(0,2,1) = 0. \]
\[ f_{4}(0,2,2) = 0. \]
\[ f_{4}(1,0,0) = 0. \]
\[ f_{4}(1,0,1) = 0. \]
\[ f_{4}(1,0,2) = 0. \]
\[ f_{4}(1,1,0) = 0. \]
\[ f_{4}(1,1,1) = 0. \]
\[ f_{4}(1,1,2) = 0. \]
\[ f_{4}(1,2,0) = 0. \]
\[ f_{4}(1,2,1) = 0. \]
\[ f_{4}(1,2,2) = 0. \]
\[ f_{4}(2,0,0) = 0. \]
\[ f_{4}(2,0,1) = 0. \]
\[ f_{4}(2,0,2) = 0. \]
\[ f_{4}(2,1,0) = 0. \]
\[ f_{4}(2,1,1) = 0. \]
\[ f_{4}(2,1,2) = 0. \]
\[ f_{4}(2,2,0) = 0. \]
\[ f_{4}(2,2,1) = 0. \]
\[ f_{4}(2,2,2) = 0. \]

\[ f_{13}(0,0,0) = 0. \]
\[ f_{13}(0,0,1) = 0. \]
\[ f_{13}(0,0,2) = 0. \]
\[ f_{13}(0,1,0) = 0. \]
\[ f_{13}(0,1,1) = 0. \]
\[ f_{13}(0,1,2) = 0. \]
\begin{align*}
  f_{13}(0,2,0) &= 0. \\
  f_{13}(0,2,1) &= 0. \\
  f_{13}(0,2,2) &= 0. \\
  f_{13}(1,0,0) &= 0. \\
  f_{13}(1,0,1) &= 0. \\
  f_{13}(1,0,2) &= 0. \\
  f_{13}(1,1,0) &= 0. \\
  f_{13}(1,1,1) &= 0. \\
  f_{13}(1,1,2) &= 0. \\
  f_{13}(1,2,0) &= 0. \\
  f_{13}(1,2,1) &= 0. \\
  f_{13}(1,2,2) &= 0. \\
  f_{13}(2,0,0) &= 0. \\
  f_{13}(2,0,1) &= 0. \\
  f_{13}(2,0,2) &= 0. \\
  f_{13}(2,1,0) &= 0. \\
  f_{13}(2,1,1) &= 0. \\
  f_{13}(2,1,2) &= 0. \\
  f_{13}(2,2,0) &= 0. \\
  f_{13}(2,2,1) &= 0. \\
  f_{13}(2,2,2) &= 0. \\
  f_{15}(0,0,0) &= 0. \\
  f_{15}(0,0,1) &= 0. \\
  f_{15}(0,0,2) &= 0. \\
  f_{15}(0,1,0) &= 0. \\
  f_{15}(0,1,1) &= 0. \\
  f_{15}(0,1,2) &= 0. \\
  f_{15}(0,2,0) &= 0. \\
  f_{15}(0,2,1) &= 0. \\
  f_{15}(0,2,2) &= 0. \\
  f_{15}(1,0,0) &= 0. \\
  f_{15}(1,0,1) &= 0. \\
  f_{15}(1,0,2) &= 0. \\
  f_{15}(1,1,0) &= 0. \\
  f_{15}(1,1,1) &= 0. \\
  f_{15}(1,1,2) &= 0. \\
  f_{15}(1,2,0) &= 0. \\
  f_{15}(1,2,1) &= 0. \\
  f_{15}(1,2,2) &= 0. \\
  f_{15}(2,0,0) &= 0. \\
  f_{15}(2,0,1) &= 0. \\
  f_{15}(2,0,2) &= 0.
\end{align*}
\[ f_{15}(2,1,0) = 0. \]
\[ f_{15}(2,1,1) = 0. \]
\[ f_{15}(2,1,2) = 0. \]
\[ f_{15}(2,2,0) = 0. \]
\[ f_{15}(2,2,1) = 0. \]
\[ f_{15}(2,2,2) = 0. \]

- AB(0).
  - AB(1).
  - AB(2).

- ACC(0).
  - ACC(1).
  - ACC(2).

- ACH(0).
  - ACH(1).
  - ACH(2).

- APO(0).
  - APO(1).
  - APO(2).

- AQ(0).
  - AQ(1).
  - AQ(2).

- AR(0).
  - AR(1).
  - AR(2).

- AS(0).
  - AS(1).
  - AS(2).

- ASO(0).
  - ASO(1).
  - ASO(2).

- AtP(0).
  - AtP(1).
  - AtP(2).
- ED(0).
- ED(1).
- ED(2).

- EV(0).
- EV(1).
- EV(2).

- F(0).
- F(1).
- F(2).

- M(0).
- M(1).
- M(2).

- MOB(0).
- MOB(1).
- MOB(2).

- NAPO(0).
- NAPO(1).
- NAPO(2).

- NASO(0).
- NASO(1).
- NASO(2).

- NPED(0).
- NPED(1).
- NPED(2).

- NPOB(0).
- NPOB(1).
- NPOB(2).

- PD(0).
- PD(1).
- PD(2).

- PED(0).
- PED(1).
- PED(2).
- POB(0).
- POB(1).
- POB(2).

- PQ(0).
- PQ(1).
- PQ(2).

- PR(0).
- PR(1).
- PR(2).

- PRO(0).
- PRO(1).
- PRO(2).

PT(0).
PT(1).
PT(2).

- Q(0).
- Q(1).
- Q(2).

- R(0).
- R(1).
- R(2).

- S(0).
- S(1).
- S(2).

- SAG(0).
- SAG(1).
- SAG(2).

- SC(0).
- SC(1).
- SC(2).

- SL(0).
- SL(1).
- SL(2).
- SOB(0).
- SOB(1).
- SOB(2).

- ST(0).
- ST(1).
- ST(2).

- STV(0).
- STV(1).
- STV(2).

- T(0).
- T(1).
- T(2).

- TL(0).
- TL(1).
- TL(2).

- TQ(0).
- TQ(1).
- TQ(2).

- TR(0).
- TR(1).
- TR(2).

- g_Event(0).
- g_Event(1).
- g_Event(2).

- g_Object(0).
- g_Object(1).
- g_Object(2).

- tr_event(0).
- tr_event(1).
- tr_event(2).

- tr_object(0).
- tr_object(1).
- tr_object(2).

- DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).

- O(0,0).
- O(0,1).
- O(0,2).
- O(1,0).
- O(1,1).
- O(1,2).
- O(2,0).
- O(2,1).
- O(2,2).

- P(0,0).
- P(0,1).
- P(0,2).
- P(1,0).
- P(1,1).
- P(1,2).
- P(2,0).
- P(2,1).
- P(2,2).

- PP(0,0).
- PP(0,1).
- PP(0,2).
- PP(1,0).
- PP(1,1).
- PP(1,2).
- PP(2,0).
- PP(2,1).
- PP(2,2).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- PRE(0,0).
  - PRE(0,1).
  - PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- PRE(1,2).
- PRE(2,0).
  - PRE(2,1).
  - PRE(2,2).

- U(0,0).
  - U(0,1).
  - U(0,2).
  - U(1,0).
  - U(1,1).
  - U(1,2).
  - U(2,0).
  - U(2,1).
  - U(2,2).

- g_hasParticipant(0,0).
  - g_hasParticipant(0,1).
  - g_hasParticipant(0,2).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).
- g_hasParticipant(1,2).
- g_hasParticipant(2,0).
- g_hasParticipant(2,1).
- g_hasParticipant(2,2).

- g_pre(0,0).
  - g_pre(0,1).
  - g_pre(0,2).
  - g_pre(1,0).
  - g_pre(1,1).
  - g_pre(1,2).
  - g_pre(2,0).
  - g_pre(2,1).
  - g_pre(2,2).

- tr_exists_at(0,0).
  - tr_exists_at(0,1).
  - tr_exists_at(0,2).
Appendix B. Comparing Participation Ontologies: Proof Files

- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,0,2).
- PC(0,1,0).
- PC(0,1,1).
- PC(0,1,2).
- PC(0,2,0).
- PC(0,2,1).
- PC(0,2,2).
- PC(1,0,0).
- PC(1,0,1).
- PC(1,0,2).
- PC(1,1,0).
- PC(1,1,1).
- PC(1,1,2).
- PC(1,2,0).
- PC(1,2,1).
- PC(1,2,2).
- PC(2,0,0).
- PC(2,0,1).
- PC(2,0,2).
- PC(2,1,0).
- PC(2,1,1).
- PC(2,1,2).
- PC(2,2,0).
- PC(2,2,1).
- \( PC(2,2,2) \).

- \( SUM(0,0,0) \).
- \( SUM(0,0,1) \).
- \( SUM(0,0,2) \).
- \( SUM(0,1,0) \).
- \( SUM(0,1,1) \).
- \( SUM(0,1,2) \).
- \( SUM(0,2,0) \).
- \( SUM(0,2,1) \).
- \( SUM(0,2,2) \).
- \( SUM(1,0,0) \).
- \( SUM(1,0,1) \).
- \( SUM(1,0,2) \).
- \( SUM(1,1,0) \).
- \( SUM(1,1,1) \).
- \( SUM(1,1,2) \).
- \( SUM(1,2,0) \).
- \( SUM(1,2,1) \).
- \( SUM(1,2,2) \).
- \( SUM(2,0,0) \).
- \( SUM(2,0,1) \).
- \( SUM(2,0,2) \).
- \( SUM(2,1,0) \).
- \( SUM(2,1,1) \).
- \( SUM(2,1,2) \).
- \( SUM(2,2,0) \).
- \( SUM(2,2,1) \).
- \( SUM(2,2,2) \).

- \( tr\_active\_in(0,0,0) \).
- \( tr\_active\_in(0,0,1) \).
- \( tr\_active\_in(0,0,2) \).
- \( tr\_active\_in(0,1,0) \).
- \( tr\_active\_in(0,1,1) \).
- \( tr\_active\_in(0,1,2) \).
- \( tr\_active\_in(0,2,0) \).
- \( tr\_active\_in(0,2,1) \).
- \( tr\_active\_in(0,2,2) \).
- \( tr\_active\_in(1,0,0) \).
- \( tr\_active\_in(1,0,1) \).
- \( tr\_active\_in(1,0,2) \).
- \( tr\_active\_in(1,1,0) \).
- `tr_active_in(1,1,1)`.  
- `tr_active_in(1,1,2)`.  
- `tr_active_in(1,2,0)`.  
- `tr_active_in(1,2,1)`.  
- `tr_active_in(1,2,2)`.  
- `tr_active_in(2,0,0)`.  
- `tr_active_in(2,0,1)`.  
- `tr_active_in(2,0,2)`.  
- `tr_active_in(2,1,0)`.  
- `tr_active_in(2,1,1)`.  
- `tr_active_in(2,1,2)`.  
- `tr_active_in(2,2,0)`.  
- `tr_active_in(2,2,1)`.  
- `tr_active_in(2,2,2)`.  

- `tr_consumed_in(0,0,0)`.  
- `tr_consumed_in(0,0,1)`.  
- `tr_consumed_in(0,0,2)`.  
- `tr_consumed_in(0,1,0)`.  
- `tr_consumed_in(0,1,1)`.  
- `tr_consumed_in(0,1,2)`.  
- `tr_consumed_in(0,2,0)`.  
- `tr_consumed_in(0,2,1)`.  
- `tr_consumed_in(0,2,2)`.  
- `tr_consumed_in(1,0,0)`.  
- `tr_consumed_in(1,0,1)`.  
- `tr_consumed_in(1,0,2)`.  
- `tr_consumed_in(1,1,0)`.  
- `tr_consumed_in(1,1,1)`.  
- `tr_consumed_in(1,1,2)`.  
- `tr_consumed_in(1,2,0)`.  
- `tr_consumed_in(1,2,1)`.  
- `tr_consumed_in(1,2,2)`.  
- `tr_consumed_in(2,0,0)`.  
- `tr_consumed_in(2,0,1)`.  
- `tr_consumed_in(2,0,2)`.  
- `tr_consumed_in(2,1,0)`.  
- `tr_consumed_in(2,1,1)`.  
- `tr_consumed_in(2,1,2)`.  
- `tr_consumed_in(2,2,0)`.  
- `tr_consumed_in(2,2,1)`.  
- `tr_consumed_in(2,2,2)`. 
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- tr_has_role(0,0,0).
- tr_has_role(0,0,1).
- tr_has_role(0,0,2).
- tr_has_role(0,1,0).
- tr_has_role(0,1,1).
- tr_has_role(0,1,2).
- tr_has_role(0,2,0).
- tr_has_role(0,2,1).
- tr_has_role(0,2,2).
- tr_has_role(1,0,0).
- tr_has_role(1,0,1).
- tr_has_role(1,0,2).
- tr_has_role(1,1,0).
- tr_has_role(1,1,1).
- tr_has_role(1,1,2).
- tr_has_role(1,2,0).
- tr_has_role(1,2,1).
- tr_has_role(1,2,2).
- tr_has_role(2,0,0).
- tr_has_role(2,0,1).
- tr_has_role(2,0,2).
- tr_has_role(2,1,0).
- tr_has_role(2,1,1).
- tr_has_role(2,1,2).
- tr_has_role(2,2,0).
- tr_has_role(2,2,1).
- tr_has_role(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
  tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

- tr_passive_in(0,0,0).
- tr_passive_in(0,0,1).
- tr_passive_in(0,0,2).
- tr_passive_in(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(0,1,2).
- tr_passive_in(0,2,0).
- tr_passive_in(0,2,1).
- tr_passive_in(0,2,2).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,0,2).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).
- tr_passive_in(1,1,2).
- tr_passive_in(1,2,0).
- tr_passive_in(1,2,1).
- tr_passive_in(1,2,2).
- tr_passive_in(2,0,0).
  tr_passive_in(2,0,1).
- tr_passive_in(2,0,2).
- tr_passive_in(2,1,0).
- tr_passive_in(2,1,1).
- tr_passive_in(2,1,2).
- tr_passive_in(2,2,0).
- tr_passive_in(2,2,1).
- tr_passive_in(2,2,2).
B.3.6  \( T_i = \text{gangemi\_participation}, T_j = \text{DOLCE\_participation} \)

\( \text{gangemi\_participation} \cup T_R \cup \models \text{DOLCE\_participation} \)

Proof attempt for axiom 1 of DOLCE\_participation:

\[
\begin{align*}
\text{p9} & \quad \text{(all x (g\_Event(x) \rightarrow (exists y (g\_Object(x) & g\_hasParticipant(x,y))))).} \\
\text{p9} & \quad \text{(all x (g\_Object(x) \rightarrow (exists y (g\_Event(x) & g\_hasParticipant(y,x))))).} \\
\text{p9} & \quad \text{(all x (g\_Event(x) \rightarrow (exists y g\_pre(x,y))))}. \\
\text{p9} & \quad \text{(all x (g\_Object(x) \rightarrow (exists y g\_pre(x,y))))}. \\
\text{p9} & \quad \text{(all x all y (g\_pre(x,y) \rightarrow (g\_Object(x) | g\_Event(x)) & g\_Time(y))).} \\
\end{align*}
\]

\[
\begin{align*}
\text{p9} & \quad \text{(all x (g\_Event(x) <-> tr\_event(x))).} \\
\text{p9} & \quad \text{(all x (g\_Object(x) <-> tr\_object(x))).} \\
\text{p9} & \quad \text{(all x all y (g\_hasParticipant(x,y) <-> (exists t (tr\_participates\_in(y,x,t))))).} \\
\text{p9} & \quad \text{(all x all y (g\_pre(x,y) <-> tr\_exists\_at(x,t))).} \\
\end{align*}
\]

\[
\begin{align*}
\text{p9} & \quad \text{%'PC equivalent to tr\_participates\_in'} \\
\text{p9} & \quad \text{(all x all a all t (PC(x,a,t) <-> tr\_participates\_in(x,a,t))).} \\
\end{align*}
\]

\[
\begin{align*}
\text{p9} & \quad \text{%'PD equivalent to tr\_event'} \\
\text{p9} & \quad \text{(all a (PD(a) <-> tr\_event(a))).} \\
\end{align*}
\]

\[
\begin{align*}
\text{p9} & \quad \text{%'no dolce\_participation equivalent for tr\_has\_role'} \\
\text{p9} & \quad \text{%'PRE equivalent to tr\_exists\_at'} \\
\text{p9} & \quad \text{(all a all t (PRE(a,t) <-> tr\_exists\_at(a,t))).} \\
\text{p9} & \quad \text{%'no dolce\_participation equivalent for tr\_part\_of'} \\
\end{align*}
\]
Appendix B. Comparing Participation Ontologies: Proof Files

% 'no dolce_participation equivalent for tr_object but we are given the
  additional info that an object is an endurant'
%(all a (tr_object(a) -> ED(a))).

% 'no dolce_participation equivalent for tr_active_in'
% 'no dolce_participation equivalent for tr_passive_in'
% 'no dolce_participation equivalent for tr_consumed_in'

%root theory: % Root Theory (collection of candidates' signature term
tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)
  ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,
  occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (FR(x) -> FR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%Tr
%CQ1 'Anything that participates in some event has some role when they are participating.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
   (exists z
    tr_has_role(x,z,t))))).

%CQ2 'An object must exist in order for it to participate in some event.'
(all x all a all t
  (tr_participates_in(x,a,t)
   & tr_event(a)
   ->
    tr_exists_at(x,t))).
%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
(all x all a_1 all a_2 all t
(tr_participates_in(x,a_1,t)
& tr_part_of(a_1,a_2)
->
tr_participates_in(x,a_2,t)))).

%CQ4 'People can participate in more than one activity, but objects can not.'
(all x all a_1 all a_2
(tr_object(x)
& tr_participates_in(x,a_1,t)
->
((a_1 = a_2) | tr_part_of(a_1,a_2)))).

%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
(tr_participates_in(x,a,t)
->
(tr_active_in(x,a,t) | tr_passive_in(x,a,t) |
tr_consumed_in(x,a,t)))).

%goal
%ax1
-((all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t)))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.
c1 = 0.
c2 = 0.
c3 = 0.
psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.

f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0,0) = 0.
f5(0,1) = 0.
f5(1,0) = 0.
f5(1,1) = 0.

f6(0,0,0) = 0.
f6(0,0,1) = 0.
f6(0,1,0) = 0.
f6(0,1,1) = 0.
f6(1,0,0) = 0.
f6(1,0,1) = 0.
f6(1,1,0) = 0.
f6(1,1,1) = 0.

- ED(0).
- ED(1).

- PD(0).
- PD(1).

- T(0).
- T(1).

- g_Event(0).
- g_Event(1).
- g_Object(0).
- g_Object(1).

- g_Time(0).
- g_Time(1).

- tr_event(0).
- tr_event(1).

- tr_object(0).
- tr_object(1).

- PRE(0,0).
- PRE(0,1).
- PRE(1,0).
- PRE(1,1).

- g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).

- g_pre(0,0).
- g_pre(0,1).
- g_pre(1,0).
- g_pre(1,1).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,1,0).
- PC(0,1,1).
- PC(1,0,0).
- \text{PC}(1,0,1).
- \text{PC}(1,1,0).
- \text{PC}(1,1,1).

- \text{tr\_active\_in}(0,0,0).
- \text{tr\_active\_in}(0,0,1).
- \text{tr\_active\_in}(0,1,0).
- \text{tr\_active\_in}(0,1,1).
- \text{tr\_active\_in}(1,0,0).
- \text{tr\_active\_in}(1,0,1).
- \text{tr\_active\_in}(1,1,0).
- \text{tr\_active\_in}(1,1,1).

- \text{tr\_consumed\_in}(0,0,0).
- \text{tr\_consumed\_in}(0,0,1).
- \text{tr\_consumed\_in}(0,1,0).
- \text{tr\_consumed\_in}(0,1,1).
- \text{tr\_consumed\_in}(1,0,0).
- \text{tr\_consumed\_in}(1,0,1).
- \text{tr\_consumed\_in}(1,1,0).
- \text{tr\_consumed\_in}(1,1,1).

- \text{tr\_has\_role}(0,0,0).
- \text{tr\_has\_role}(0,0,1).
- \text{tr\_has\_role}(0,1,0).
- \text{tr\_has\_role}(0,1,1).
- \text{tr\_has\_role}(1,0,0).
- \text{tr\_has\_role}(1,0,1).
- \text{tr\_has\_role}(1,1,0).
- \text{tr\_has\_role}(1,1,1).

\text{tr\_participates\_in}(0,0,0).
- \text{tr\_participates\_in}(0,0,1).
- \text{tr\_participates\_in}(0,1,0).
- \text{tr\_participates\_in}(0,1,1).
- \text{tr\_participates\_in}(1,0,0).
- \text{tr\_participates\_in}(1,0,1).
- \text{tr\_participates\_in}(1,1,0).
- \text{tr\_participates\_in}(1,1,1).

\text{tr\_passive\_in}(0,0,0).
- \text{tr\_passive\_in}(0,0,1).
- \text{tr\_passive\_in}(0,1,0).
- tr_passive_in(0,1,1).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).

## B.4 Remaining CQ Evaluation Results

This section provides the individual evaluation results of each candidate against the remainder of the CQs (as not all CQs need have been evaluated to assess the preference ordering). As in the previous section, because the mappings weren’t syntactically applied to each theory, we include the mapping axioms for any theories involved in the antecedent of a given entailment problem to attain the shared signature. Proof or model output included where applicable.

### B.4.1 H_PSL_participates

$H_{PSL}\text{-participates} \neq CQ2$ Proof attempt for CQ2:

%psl_participates for common hierarchy
%psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))).
(all x (psl_object(x) -> -psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) & psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) & psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) & psl_is_occuring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) & psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) & (psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) & psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,psl_endof(x)))).
(all occ all t (psl_is_occuring_at(occ,t) <-> psl_activity_occurrence(occ) & psl_betweenEq(beginof(occ),t,endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
% psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t  (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a  (psl_activity_occurrence(a) <-> tr_event(a))).

% 'exists_at equivalent to tr_exists_at'
(all a  (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2  (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

% Root Theory
% Root Theory (collection of candidates' signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t  (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))
(all a  (tr_event(a) -> tr_event(a))).
(all x all z all t  (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t  (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2  (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x  (tr_object(x) -> tr_object(x))).
(all x  (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x  (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x  (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

% gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x  (g_Event(x) -> g_Event(x))).
(all x  (g_Object(x) -> g_Object(x))).
(all x all y  (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y  (g_pre(x,y) -> g_pre(x,y))).
(all y  (g_Time(y) -> g_Time(y))).

% psl_participation.p9 signature
% psl_ prefix added to all terms to distinguish from other candidates
(all x  (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x  (psl_object(x) -> psl_object(x))).
(all x  (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,
,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))
.
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ2 'An object must exist in order for it to participate in some event.'
-((all x all a all t
    (tr_participates_in(x,a,t)
    & tr_event(a)
    -> tr_exists_at(x,t)))).

Counterexample found:

% number = 1
% seconds = 0
% Interpretation of size 3

t = 0.

c1 = 0.

c2 = 1.

c3 = 2.

beginof(0) = 2.
beginof(1) = 2.
beginof(2) = 0.

endof(0) = 0.
endof(1) = 2.
endof(2) = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 2.
psl_endof(1) = 0.
psl_endof(2) = 0.

- psl_activity_occurrence(0).
  - psl_activity_occurrence(1).
  - psl_activity_occurrence(2).

  psl_object(0).
  - psl_object(1).
  - psl_object(2).

- psl_timepoint(0).
  - psl_timepoint(1).
  - psl_timepoint(2).

- tr_event(0).
  tr_event(1).
  - tr_event(2).
- psl_before(0,0).
- psl_before(0,1).
- psl_before(0,2).
- psl_before(1,0).
- psl_before(1,1).
- psl_before(1,2).
- psl_before(2,0).
- psl_before(2,1).
- psl_before(2,2).

- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(0,2).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).
- psl_beforeEq(1,2).
- psl_beforeEq(2,0).
- psl_beforeEq(2,1).
- psl_beforeEq(2,2).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(0,2).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).

- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(0,2).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).
- psl_is_occuring_at(1,2).
- psl_is_occuring_at(2,0).
- psl_is_occuring_at(2,1).
- psl_is_occuring_at(2,2).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(2,0).
- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).

- psl_between(0,0,0).
- psl_between(0,0,1).
- psl_between(0,0,2).
- psl_between(0,1,0).
- psl_between(0,1,1).
- psl_between(0,1,2).
- psl_between(0,2,0).
- psl_between(0,2,1).
- psl_between(0,2,2).
- psl_between(1,0,0).
- psl_between(1,0,1).
- psl_between(1,0,2).
- psl_between(1,1,0).
- psl_between(1,1,1).
- psl_between(1,1,2).
- psl_between(1,2,0).
- psl_between(1,2,1).
- psl_between(1,2,2).
- psl_between(2,0,0).
- psl_between(2,0,1).
- psl_between(2,0,2).
- psl_between(2,1,0).
- psl_between(2,1,1).
- psl_between(2,1,2).
- psl_between(2,2,0).
- psl_between(2,2,1).
- psl_between(2,2,2).
- psl_betweenEq(0,0,0).
- psl_betweenEq(0,0,1).
- psl_betweenEq(0,0,2).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(0,1,2).
- psl_betweenEq(0,2,0).
- psl_betweenEq(0,2,1).
- psl_betweenEq(0,2,2).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,0,2).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).
- psl_betweenEq(1,1,2).
- psl_betweenEq(1,2,0).
- psl_betweenEq(1,2,1).
- psl_betweenEq(1,2,2).
- psl_betweenEq(2,0,0).
- psl_betweenEq(2,0,1).
- psl_betweenEq(2,0,2).
- psl_betweenEq(2,1,0).
- psl_betweenEq(2,1,1).
- psl_betweenEq(2,1,2).
- psl_betweenEq(2,2,0).
- psl_betweenEq(2,2,1).
- psl_betweenEq(2,2,2).
- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
  psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
  tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

\[ H_{PSL_{participates}} \neq CQ3 \] Proof attempt for CQ3:

\[
\% psl\_participates for common hierarchy \\
\% psl\_ prefix added to all terms to distinguish from other candidates
\]

\[
\begin{align*}
(\text{all } x \ (\text{psl}\_\text{activity}\_\text{occurrence}(x) & \rightarrow \neg \text{psl}\_\text{object}(x) \land \neg \text{psl}\_\text{timepoint}(x))). \\
(\text{all } x \ (\text{psl}\_\text{object}(x) & \rightarrow \neg \text{psl}\_\text{timepoint}(x))). \\
(\text{all } o \text{ all } t \ (\text{psl}\_\text{is}\_\text{occurring}\_\text{at}(o,t) & \rightarrow \text{psl}\_\text{activity}\_\text{occurrence}(o) \land \text{psl}\_\text{timepoint}(t))). \\
(\text{all } x \text{ all } t \ (\text{psl}\_\text{exists}\_\text{at}(x,t) & \rightarrow \text{psl}\_\text{object}(x) \land \text{psl}\_\text{timepoint}(t))). \\
(\text{all } x \text{ all } occ \text{ all } t \ (\text{psl}\_\text{participates}\_\text{in}(x,occ,t) & \rightarrow \text{psl}\_\text{object}(x) \land \text{psl}\_\text{activity}\_\text{occurrence}(occ) \land \text{psl}\_\text{timepoint}(t))). \\
(\text{all } x \text{ all } occ \text{ all } t \ (\text{psl}\_\text{participates}\_\text{in}(x,occ,t) & \rightarrow \text{psl}\_\text{exists}\_\text{at}(x,t) \land \text{psl}\_\text{is}\_\text{occurring}\_\text{at}(occ,t))). \\
(\text{all } t_1 \text{ all } t_2 \text{ all } t_3 \ (\text{psl}\_\text{between}(t_1,t_2,t_3) & \rightarrow \text{psl}\_\text{before}(t_1,t_2) \land \text{psl}\_\text{before}(t_2,t_3))). \\
(\text{all } t_1 \text{ all } t_2 \ (\text{psl}\_\text{beforeEq}(t_1,t_2) & \rightarrow \text{psl}\_\text{timepoint}(t_1) \land \text{psl}\_\text{timepoint}(t_2) \land (\text{psl}\_\text{before}(t_1,t_2) \land t_1 = t_2))). \\
(\text{all } t_1 \text{ all } t_2 \text{ all } t_3 \ (\text{psl}\_\text{betweenEq}(t_1,t_2,t_3) & \rightarrow \text{psl}\_\text{beforeEq}(t_1,t_2) \land \text{psl}\_\text{beforeEq}(t_2,t_3))). \\
(\text{all } x \text{ all } t \ (\text{psl}\_\text{exists}\_\text{at}(x,t) & \rightarrow \text{psl}\_\text{object}(x) \land \text{psl}\_\text{betweenEq}(\text{beginof}(x),t,\text{psl}\_\text{endof}(x)))). \\
(\text{all } occ \text{ all } t \ (\text{psl}\_\text{is}\_\text{occurring}\_\text{at}(occ,t) & \rightarrow \text{psl}\_\text{activity}\_\text{occurrence}(occ) \land \text{psl}\_\text{betweenEq}(\text{beginof}(occ),t,\text{endof}(occ)))). \\
\end{align*}
\]

\%relevance mappings to requirements
\% psl_2_Tr
\% psl\_ prefix added to all terms to distinguish from other candidates

\%
\% 'participates\_in equivalent to tr\_participates\_in'
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of (o_1,o_2))).

% root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))
  .
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
-((all x all a_1 all a_2 all t
    (tr_participates_in(x,a_1,t)
    & tr_part_of(a_1,a_2)
    ->
    tr_participates_in(x,a_2,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

  t = 0.

  c1 = 0.
c2 = 1.
c3 = 0.
c4 = 2.

beginof(0) = 2.
beginof(1) = 2.
beginof(2) = 0.

deof(0) = 0.
deof(1) = 2.
deof(2) = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 2.
psl_endof(1) = 0.
psl_endof(2) = 0.

- psl_activity_occurrence(0).
  psl_activity_occurrence(1).
- psl_activity_occurrence(2).

  psl_object(0).
- psl_object(1).
- psl_object(2).

- psl_timepoint(0).
- psl_timepoint(1).
  psl_timepoint(2).

- tr_event(0).
  tr_event(1).
- tr_event(2).

- psl_before(0,0).
- psl_before(0,1).
- psl_before(0,2).
- psl_before(1,0).
- psl_before(1,1).
- psl_before(1,2).
- psl_before(2,0).
- psl_before(2,1).
- psl_before(2,2).

- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(0,2).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).
- psl_beforeEq(1,2).
- psl_beforeEq(2,0).
- psl_beforeEq(2,1).
- psl_beforeEq(2,2).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(0,2).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).

- psl_is_occurring_at(0,0).
- psl_is_occurring_at(0,1).
- psl_is_occurring_at(0,2).
- psl_is_occurring_at(1,0).
- psl_is_occurring_at(1,1).
- psl_is_occurring_at(1,2).
- psl_is_occurring_at(2,0).
- psl_is_occurring_at(2,1).
- psl_is_occurring_at(2,2).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(2,0).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).

- psl_between(0,0,0).
- psl_between(0,0,1).
- psl_between(0,0,2).
- psl_between(0,1,0).
- psl_between(0,1,1).
- psl_between(0,1,2).
- psl_between(0,2,0).
- psl_between(0,2,1).
- psl_between(0,2,2).
- psl_between(1,0,0).
- psl_between(1,0,1).
- psl_between(1,0,2).
- psl_between(1,1,0).
- psl_between(1,1,1).
- psl_between(1,1,2).
- psl_between(1,2,0).
- psl_between(1,2,1).
- psl_between(1,2,2).
- psl_between(2,0,0).
- psl_between(2,0,1).
- psl_between(2,0,2).
- psl_between(2,1,0).
- psl_between(2,1,1).
- psl_between(2,1,2).
- psl_between(2,2,0).
- psl_between(2,2,1).
- psl_between(2,2,2).

- psl_betweenEq(0,0,0).
- psl_betweenEq(0,0,1).
- psl_betweenEq(0,0,2).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(0,1,2).
- psl_betweenEq(0,2,0).
- psl_betweenEq(0,2,1).
- psl_betweenEq(0,2,2).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,0,2).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).
- psl_betweenEq(1,1,2).
- psl_betweenEq(1,2,0).
- psl_betweenEq(1,2,1).
- psl_betweenEq(1,2,2).
- psl_betweenEq(2,0,0).
- psl_betweenEq(2,0,1).
- psl_betweenEq(2,0,2).
- psl_betweenEq(2,1,0).
- psl_betweenEq(2,1,1).
- psl_betweenEq(2,1,2).
- psl_betweenEq(2,2,0).
- psl_betweenEq(2,2,1).
- psl_betweenEq(2,2,2).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).

$H_{\text{PSL.participates}} \neq CQ4$ Proof attempt for CQ4:

%psl_participates for common hierarchy
%psl_ prefix added to all terms to distinguish from other candidates

(all x (psl_activity_occurrence(x) -> -psl_object(x) & -psl_timepoint(x))%).
(all x (psl_object(x) -> -psl_timepoint(x))%).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) &
psl_timepoint(t))%).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))%).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) &
psl_activity_occurrence(occ) & psl_timepoint(t))%).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) &
psl_is_occurring_at(occ,t))%).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) &
psl_before(t2,t3))%).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) &
psl_before(t1,t2) | t1 = t2))%).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) &
psl_beforeEq(t2,t3))%).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,psl_endof(x)))%).
(all occ all t (psl_is_occurring_at(occ,t) <-> psl_activity_occurrence(occ) &
psl_betweenEq(beginof(occ),t,psl_endof(occ)))%).

%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t)))%.

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a)))%.

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t)))%.
% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1, o_2) <-> tr_part_of (o_1, o_2))).

%root theory
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x, a, t) -> (tr_participates_in(x, a, t)))
(all a (tr_event(a) -> tr_event(a))
(all x all z all t (tr_has_role(x, z, t) -> (tr_has_role(x, z, t))))
(all x all t (tr_exists_at(x, t) -> (tr_exists_at(x, t))))
(all a_1 all a_2 (tr_part_of(a_1, a_2) -> tr_part_of(a_1, a_2))
(all x (tr_object(x) -> tr_object(x))
(all x all a all t (tr_active_in(x, a, t) -> tr_active_in(x, a, t))
(all x all a all t (tr_passive_in(x, a, t) -> tr_passive_in(x, a, t))
(all x all a all t (tr_consumed_in(x, a, t) -> tr_consumed_in(x, a, t))

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))
(all x (g_Object(x) -> g_Object(x))
(all x all y (g_hasParticipant(x, y) -> g_hasParticipant(x, y))
(all y (g_Time(y) -> g_Time(y))

%psl_participation.p9 signature
% psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))
(all x (psl_object(x) -> psl_object(x))
(all x (psl_timepoint(x) -> psl_timepoint(x))
(all o all t (psl_is_occuring_at(o, t) -> psl_is_occuring_at(o, t))
(all x all t (psl_exists_at(x, t) -> psl_exists_at(x, t))
(all x all occ all t (psl_participates_in(x, occ, t) -> psl_participates_in(x , occ, t))
(all t1 all t2 all t3 (psl_between(t1, t2, t3) -> psl_before(t1, t2, t3))
(all t1 all t2 (psl_before(t1, t2) -> psl_beforeEq(t1, t2))
(all t1 all t2 (psl_beforeEq(t1, t2) -> psl_beforeEq(t1, t2))
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))
(all t1 all t2 all t3 (psl_betweenEq(t1, t2, t3) -> psl_betweenEq(t1, t2, t3))
(all x psl_endof(x) = psl_endof(x)).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (RO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ4 'Objects can not participate in more than one different activity.'
-((all x all a_1 all a_2
   (tr_object(x)
   & tr_participates_in(x,a_1,t)
   & tr_participates_in(x,a_2,t)
   ->
   ((a_1 = a_2) | tr_part_of(a_1,a_2))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 4

t = 0.
c1 = 1.
c2 = 2.
c3 = 3.
beginof(0) = 0.
beginof(1) = 0.
beginof(2) = 0.
beginof(3) = 0.

endof(0) = 0.
endof(1) = 0.
endof(2) = 0.
endof(3) = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.
psl_beginof(3) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.
psl_endof(2) = 0.
psl_endof(3) = 0.

- psl_activity_occurrence(0).
- psl_activity_occurrence(1).
  psl_activity_occurrence(2).
  psl_activity_occurrence(3).

- psl_object(0).
  psl_object(1).
- psl_object(2).
- psl_object(3).

  psl_timepoint(0).
- psl_timepoint(1).
- psl_timepoint(2).
- psl_timepoint(3).

- tr_event(0).
- tr_event(1).
  tr_event(2).
  tr_event(3).

- tr_object(0).
  tr_object(1).
- tr_object(2).
- tr_object(3).

- psl_before(0,0).
- psl_before(0,1).
- psl_before(0,2).
- psl_before(0,3).
- psl_before(1,0).
- psl_before(1,1).
- psl_before(1,2).
- psl_before(1,3).
- psl_before(2,0).
- psl_before(2,1).
- psl_before(2,2).
- psl_before(2,3).
- psl_before(3,0).
- psl_before(3,1).
- psl_before(3,2).
- psl_before(3,3).

- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(0,2).
- psl_beforeEq(0,3).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).
- psl_beforeEq(1,2).
- psl_beforeEq(1,3).
- psl_beforeEq(2,0).
- psl_beforeEq(2,1).
- psl_beforeEq(2,2).
- psl_beforeEq(2,3).
- psl_beforeEq(3,0).
- psl_beforeEq(3,1).
- psl_beforeEq(3,2).
- psl_beforeEq(3,3).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
- psl_exists_at(0,2).
- psl_exists_at(0,3).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(1,3).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).
- psl_exists_at(2,3).
- psl_exists_at(3,0).
- psl_exists_at(3,1).
- psl_exists_at(3,2).
- psl_exists_at(3,3).
- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(0,2).
- psl_is_occuring_at(0,3).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).
- psl_is_occuring_at(1,2).
- psl_is_occuring_at(1,3).
- psl_is_occuring_at(2,0).
- psl_is_occuring_at(2,1).
- psl_is_occuring_at(2,2).
- psl_is_occuring_at(2,3).
- psl_is_occuring_at(3,0).
- psl_is_occuring_at(3,1).
- psl_is_occuring_at(3,2).
- psl_is_occuring_at(3,3).
- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(0,3).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
- psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(1,3).
- psl_subactivity_occurrence(2,0).
- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).
- psl_subactivity_occurrence(2,3).
- psl_subactivity_occurrence(3,0).
- psl_subactivity_occurrence(3,1).
- psl_subactivity_occurrence(3,2).
- psl_subactivity_occurrence(3,3).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(0,3).
  tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(1,3).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).
- tr_exists_at(2,3).
- tr_exists_at(3,0).
- tr_exists_at(3,1).
- tr_exists_at(3,2).
- tr_exists_at(3,3).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(0,3).
- tr_part_of(1,0).
- tr_part_of(1,1).
- tr_part_of(1,2).
- tr_part_of(1,3).
- tr_part_of(2,0).
- tr_part_of(2,1).
- tr_part_of(2,2).
- tr_part_of(2,3).
- tr_part_of(3,0).
- tr_part_of(3,1).
- tr_part_of(3,2).
- tr_part_of(3,3).

- psl_between(0,0,0).
- psl_between(0,0,1).
- psl_between(0,0,2).
- psl_between(0,0,3).
- psl_between(0,1,0).
- psl_between(0,1,1).
- psl_between(0,1,2).
- psl_between(0,1,3).
- psl_between(0,2,0).
- psl_between(0,2,1).
- psl_between(0,2,2).
- psl_between(0,2,3).
- psl_between(0,3,0).
- psl_between(0,3,1).
- psl_between(0,3,2).
- psl_between(0,3,3).
- psl_between(1,0,0).
- psl_between(1,0,1).
- psl_between(1,0,2).
- psl_between(1,0,3).
- psl_between(1,1,0).
- psl_between(1,1,1).
- psl_between(1,1,2).
- psl_between(1,1,3).
- psl_between(1,2,0).
- psl_between(1,2,1).
- psl_between(1,2,2).
- psl_between(1,2,3).
- psl_between(1,3,0).
- psl_between(1,3,1).
- psl_between(1,3,2).
- psl_between(1,3,3).
- psl_between(2,0,0).
- psl_between(2,0,1).
- psl_between(2,0,2).
- psl_between(2,0,3).
- psl_between(2,1,0).
- psl_between(2,1,1).
- psl_between(2,1,2).
- psl_between(2,1,3).
- psl_between(2,2,0).
- psl_between(2,2,1).
- psl_between(2,2,2).
- psl_between(2,2,3).
- psl_between(2,3,0).
- psl_between(2,3,1).
- psl_between(2,3,2).
- psl_between(2,3,3).
- psl_between(3,0,0).
- psl_between(3,0,1).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- psl_between(3,0,2).
- psl_between(3,0,3).
- psl_between(3,1,0).
- psl_between(3,1,1).
- psl_between(3,1,2).
- psl_between(3,1,3).
- psl_between(3,2,0).
- psl_between(3,2,1).
- psl_between(3,2,2).
- psl_between(3,2,3).
- psl_between(3,3,0).
- psl_between(3,3,1).
- psl_between(3,3,2).
- psl_between(3,3,3).

  psl_betweenEq(0,0,0).
- psl_betweenEq(0,0,1).
- psl_betweenEq(0,0,2).
- psl_betweenEq(0,0,3).
- psl_betweenEq(0,1,0).
- psl_betweenEq(0,1,1).
- psl_betweenEq(0,1,2).
- psl_betweenEq(0,1,3).
- psl_betweenEq(0,2,0).
- psl_betweenEq(0,2,1).
- psl_betweenEq(0,2,2).
- psl_betweenEq(0,2,3).
- psl_betweenEq(0,3,0).
- psl_betweenEq(0,3,1).
- psl_betweenEq(0,3,2).
- psl_betweenEq(0,3,3).
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,0,2).
- psl_betweenEq(1,0,3).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).
- psl_betweenEq(1,1,2).
- psl_betweenEq(1,1,3).
- psl_betweenEq(1,2,0).
- psl_betweenEq(1,2,1).
- psl_betweenEq(1,2,2).
- psl_betweenEq(1,2,3).
- psl_betweenEq(1,3,0).
- psl_betweenEq(1,3,1).
- psl_betweenEq(1,3,2).
- psl_betweenEq(1,3,3).
- psl_betweenEq(2,0,0).
- psl_betweenEq(2,0,1).
- psl_betweenEq(2,0,2).
- psl_betweenEq(2,0,3).
- psl_betweenEq(2,1,0).
- psl_betweenEq(2,1,1).
- psl_betweenEq(2,1,2).
- psl_betweenEq(2,1,3).
- psl_betweenEq(2,2,0).
- psl_betweenEq(2,2,1).
- psl_betweenEq(2,2,2).
- psl_betweenEq(2,2,3).
- psl_betweenEq(2,3,0).
- psl_betweenEq(2,3,1).
- psl_betweenEq(2,3,2).
- psl_betweenEq(2,3,3).
- psl_betweenEq(3,0,0).
- psl_betweenEq(3,0,1).
- psl_betweenEq(3,0,2).
- psl_betweenEq(3,0,3).
- psl_betweenEq(3,1,0).
- psl_betweenEq(3,1,1).
- psl_betweenEq(3,1,2).
- psl_betweenEq(3,1,3).
- psl_betweenEq(3,2,0).
- psl_betweenEq(3,2,1).
- psl_betweenEq(3,2,2).
- psl_betweenEq(3,2,3).
- psl_betweenEq(3,3,0).
- psl_betweenEq(3,3,1).
- psl_betweenEq(3,3,2).
- psl_betweenEq(3,3,3).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,0,3).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,1,3).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(0,2,3).
- psl_participates_in(0,3,0).
- psl_participates_in(0,3,1).
- psl_participates_in(0,3,2).
- psl_participates_in(0,3,3).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,0,3).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,1,3).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(1,2,3).
- psl_participates_in(1,3,0).
- psl_participates_in(1,3,1).
- psl_participates_in(1,3,2).
- psl_participates_in(1,3,3).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,0,3).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,1,3).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).
- psl_participates_in(2,2,3).
- psl_participates_in(2,3,0).
- psl_participates_in(2,3,1).
- psl_participates_in(2,3,2).
- psl_participates_in(2,3,3).
- psl_participates_in(3,0,0).
- psl_participates_in(3,0,1).
- psl_participates_in(3,0,2).
- psl_participates_in(3,0,3).
- psl_participates_in(3,1,0).
- psl_participates_in(3,1,1).
- psl_participates_in(3,1,2).
- psl_participates_in(3,1,3).
- psl_participates_in(3,2,0).
- psl_participates_in(3,2,1).
- psl_participates_in(3,2,2).
- psl_participates_in(3,2,3).
- psl_participates_in(3,3,0).
- psl_participates_in(3,3,1).
- psl_participates_in(3,3,2).
- psl_participates_in(3,3,3).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,0,3).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,1,3).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(0,2,3).
- tr_participates_in(0,3,0).
- tr_participates_in(0,3,1).
- tr_participates_in(0,3,2).
- tr_participates_in(0,3,3).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,0,3).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,1,3).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(1,2,3).
  tr_participates_in(1,3,0).
- tr_participates_in(1,3,1).
- tr_participates_in(1,3,2).
- tr_participates_in(1,3,3).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,0,3).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,1,3).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).
- tr_participates_in(2,2,3).
- tr_participates_in(2,3,0).
- tr_participates_in(2,3,1).
- tr_participates_in(2,3,2).
- tr_participates_in(2,3,3).
- tr_participates_in(3,0,0).
- tr_participates_in(3,0,1).
- tr_participates_in(3,0,2).
- tr_participates_in(3,0,3).
- tr_participates_in(3,1,0).
- tr_participates_in(3,1,1).
- tr_participates_in(3,1,2).
- tr_participates_in(3,1,3).
- tr_participates_in(3,2,0).
- tr_participates_in(3,2,1).
- tr_participates_in(3,2,2).
- tr_participates_in(3,2,3).
- tr_participates_in(3,3,0).
- tr_participates_in(3,3,1).
- tr_participates_in(3,3,2).
- tr_participates_in(3,3,3).

\textit{H.PSL\_participates }\neq \textit{CQ5 Proof attempt for CQ5:}

\%psl\_participates for common hierarchy
\%psl\_ prefix added to all terms to distinguish from other candidates

\(\text{\forall x \ (psl\_activity\_occurrence}(x) \rightarrow \neg \text{psl\_object}(x) \& \neg \text{psl\_timepoint}(x))\).

(all x (psl_object(x) -> -psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_activity_occurrence(o) &
psl_timepoint(t))).
(all x all t (psl_exists_at(x,t) -> psl_object(x) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_object(x) &
psl_activity_occurrence(occ) & psl_timepoint(t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_exists_at(x,t) &
psl_is_occurring_at(occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) <-> psl_before(t1,t2) &
psl_before(t2,t3))).
(all t1 all t2 (psl_beforeEq(t1,t2) <-> psl_timepoint(t1) & psl_timepoint(t2) &
(psl_before(t1,t2) | t1 = t2))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) <-> psl_beforeEq(t1,t2) &
psl_beforeEq(t2,t3))).
(all x all t (psl_exists_at(x,t) <-> psl_object(x) & psl_betweenEq(beginof(x),t,endof(x)))).
(all occ all t (psl_is_occurring_at(occ,t) <-> psl_activity_occurrence(occ) & psl_betweenEq(beginof(occ),t,endof(occ)))).

%relevance mappings to requirements
%psl_2_Tr
%psl_ prefix added to all terms to distinguish from other candidates

% 'participates_in equivalent to tr_participates_in'
(all x all a all t (psl_participates_in(x,a,t) <-> tr_participates_in(x,a,t))).

% 'activity_occurrence equivalent to tr_event'
(all a (psl_activity_occurrence(a) <-> tr_event(a))).

% 'exists_at equivalent to tr_exists_at'
(all a (psl_exists_at(a,t) <-> tr_exists_at(a,t))).

% 'subactivity_occurrence equivalent to tr_part_of'
(all o_1 all o_2 (psl_subactivity_occurrence(o_1,o_2) <-> tr_part_of(o_1,o_2))).

%root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) &
tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2)).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NFOB(x) -> NFOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
- (%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
(all x all a all t
  (tr_participates_in(x,a,t)
   ->
   (tr_active_in(x,a,t) | tr_passive_in(x,a,t) |
    tr_consumed_in(x,a,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

t = 0.
c1 = 0.
c2 = 1.
c3 = 2.
beginof(0) = 2.
beginof(1) = 2.
beginof(2) = 0.
endof(0) = 0.
endof(1) = 2.
endof(2) = 0.
psl_beginof(0) = 0.
psl_beginof(1) = 0.
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

psl_beginof(2) = 0.

psl_endof(0) = 2.
psl_endof(1) = 0.
psl_endof(2) = 0.

- psl_activity_occurrence(0).
  psl_activity_occurrence(1).
- psl_activity_occurrence(2).

  psl_object(0).
- psl_object(1).
- psl_object(2).

  psl_timepoint(0).
- psl_timepoint(1).
  psl_timepoint(2).

  tr_event(0).
- tr_event(1).
- tr_event(2).

- psl_before(0,0).
- psl_before(0,1).
- psl_before(0,2).
- psl_before(1,0).
- psl_before(1,1).
- psl_before(1,2).
- psl_before(2,0).
- psl_before(2,1).
- psl_before(2,2).

- psl_beforeEq(0,0).
- psl_beforeEq(0,1).
- psl_beforeEq(0,2).
- psl_beforeEq(1,0).
- psl_beforeEq(1,1).
- psl_beforeEq(1,2).
- psl_beforeEq(2,0).
- psl_beforeEq(2,1).
  psl_beforeEq(2,2).

- psl_exists_at(0,0).
- psl_exists_at(0,1).
  psl_exists_at(0,2).
- psl_exists_at(1,0).
- psl_exists_at(1,1).
- psl_exists_at(1,2).
- psl_exists_at(2,0).
- psl_exists_at(2,1).
- psl_exists_at(2,2).

- psl_is_occuring_at(0,0).
- psl_is_occuring_at(0,1).
- psl_is_occuring_at(0,2).
- psl_is_occuring_at(1,0).
- psl_is_occuring_at(1,1).
  psl_is_occuring_at(1,2).
- psl_is_occuring_at(2,0).
- psl_is_occuring_at(2,1).
- psl_is_occuring_at(2,2).

- psl_subactivity_occurrence(0,0).
- psl_subactivity_occurrence(0,1).
- psl_subactivity_occurrence(0,2).
- psl_subactivity_occurrence(1,0).
- psl_subactivity_occurrence(1,1).
  psl_subactivity_occurrence(1,2).
- psl_subactivity_occurrence(2,0).
- psl_subactivity_occurrence(2,1).
- psl_subactivity_occurrence(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
- tr_part_of(1,0).
- `tr_part_of(1,1).`
- `tr_part_of(1,2).`
- `tr_part_of(2,0).`
- `tr_part_of(2,1).`
- `tr_part_of(2,2).`

- `psl_between(0,0,0).`
- `psl_between(0,0,1).`
- `psl_between(0,0,2).`
- `psl_between(0,1,0).`
- `psl_between(0,1,1).`
- `psl_between(0,1,2).`
- `psl_between(0,2,0).`
- `psl_between(0,2,1).`
- `psl_between(0,2,2).`
- `psl_between(1,0,0).`
- `psl_between(1,0,1).`
- `psl_between(1,0,2).`
- `psl_between(1,1,0).`
- `psl_between(1,1,1).`
- `psl_between(1,1,2).`
- `psl_between(1,2,0).`
- `psl_between(1,2,1).`
- `psl_between(1,2,2).`
- `psl_between(2,0,0).`
- `psl_between(2,0,1).`
- `psl_between(2,0,2).`
- `psl_between(2,1,0).`
- `psl_between(2,1,1).`
- `psl_between(2,1,2).`
- `psl_between(2,2,0).`
- `psl_between(2,2,1).`
- `psl_between(2,2,2).`

- `psl_betweenEq(0,0,0).`
- `psl_betweenEq(0,0,1).`
- `psl_betweenEq(0,0,2).`
- `psl_betweenEq(0,1,0).`
- `psl_betweenEq(0,1,1).`
- `psl_betweenEq(0,1,2).`
- `psl_betweenEq(0,2,0).`
- `psl_betweenEq(0,2,1).`
- `psl_betweenEq(0,2,2).`
- psl_betweenEq(1,0,0).
- psl_betweenEq(1,0,1).
- psl_betweenEq(1,0,2).
- psl_betweenEq(1,1,0).
- psl_betweenEq(1,1,1).
- psl_betweenEq(1,1,2).
- psl_betweenEq(1,2,0).
- psl_betweenEq(1,2,1).
- psl_betweenEq(1,2,2).
- psl_betweenEq(2,0,0).
- psl_betweenEq(2,0,1).
- psl_betweenEq(2,0,2).
- psl_betweenEq(2,1,0).
- psl_betweenEq(2,1,1).
- psl_betweenEq(2,1,2).
- psl_betweenEq(2,2,0).
- psl_betweenEq(2,2,1).
- psl_betweenEq(2,2,2).

- psl_participates_in(0,0,0).
- psl_participates_in(0,0,1).
- psl_participates_in(0,0,2).
- psl_participates_in(0,1,0).
- psl_participates_in(0,1,1).
- psl_participates_in(0,1,2).
- psl_participates_in(0,2,0).
- psl_participates_in(0,2,1).
- psl_participates_in(0,2,2).
- psl_participates_in(1,0,0).
- psl_participates_in(1,0,1).
- psl_participates_in(1,0,2).
- psl_participates_in(1,1,0).
- psl_participates_in(1,1,1).
- psl_participates_in(1,1,2).
- psl_participates_in(1,2,0).
- psl_participates_in(1,2,1).
- psl_participates_in(1,2,2).
- psl_participates_in(2,0,0).
- psl_participates_in(2,0,1).
- psl_participates_in(2,0,2).
- psl_participates_in(2,1,0).
- psl_participates_in(2,1,1).
- psl_participates_in(2,1,2).
- psl_participates_in(2,2,0).
- psl_participates_in(2,2,1).
- psl_participates_in(2,2,2).

- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,0,2).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(0,1,2).
- tr_active_in(0,2,0).
- tr_active_in(0,2,1).
- tr_active_in(0,2,2).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,0,2).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).
- tr_active_in(1,1,2).
- tr_active_in(1,2,0).
- tr_active_in(1,2,1).
- tr_active_in(1,2,2).
- tr_active_in(2,0,0).
- tr_active_in(2,0,1).
- tr_active_in(2,0,2).
- tr_active_in(2,1,0).
- tr_active_in(2,1,1).
- tr_active_in(2,1,2).
- tr_active_in(2,2,0).
- tr_active_in(2,2,1).
- tr_active_in(2,2,2).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,0,2).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(0,1,2).
- tr_consumed_in(0,2,0).
- tr_consumed_in(0,2,1).
- tr_consumed_in(0,2,2).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,0,2).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).
- tr_consumed_in(1,1,2).
- tr_consumed_in(1,2,0).
- tr_consumed_in(1,2,1).
- tr_consumed_in(1,2,2).
- tr_consumed_in(2,0,0).
- tr_consumed_in(2,0,1).
- tr_consumed_in(2,0,2).
- tr_consumed_in(2,1,0).
- tr_consumed_in(2,1,1).
- tr_consumed_in(2,1,2).
- tr_consumed_in(2,2,0).
- tr_consumed_in(2,2,1).
- tr_consumed_in(2,2,2).
- tr_consumed_in(1,0,2).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).
- tr_consumed_in(1,1,2).
- tr_consumed_in(1,2,0).
- tr_consumed_in(1,2,1).
- tr_consumed_in(1,2,2).
- tr_consumed_in(2,0,0).
- tr_consumed_in(2,0,1).
- tr_consumed_in(2,0,2).
- tr_consumed_in(2,1,0).
- tr_consumed_in(2,1,1).
- tr_consumed_in(2,1,2).
- tr_consumed_in(2,2,0).
- tr_consumed_in(2,2,1).
- tr_consumed_in(2,2,2).

- tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
- tr_participates_in(0,0,2).
- tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(0,1,2).
- tr_participates_in(0,2,0).
- tr_participates_in(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- \text{tr\_participates\_in}(2,2,2).

- \text{tr\_passive\_in}(0,0,0).
- \text{tr\_passive\_in}(0,0,1).
- \text{tr\_passive\_in}(0,0,2).
- \text{tr\_passive\_in}(0,1,0).
- \text{tr\_passive\_in}(0,1,1).
- \text{tr\_passive\_in}(0,1,2).
- \text{tr\_passive\_in}(0,2,0).
- \text{tr\_passive\_in}(0,2,1).
- \text{tr\_passive\_in}(0,2,2).
- \text{tr\_passive\_in}(1,0,0).
- \text{tr\_passive\_in}(1,0,1).
- \text{tr\_passive\_in}(1,0,2).
- \text{tr\_passive\_in}(1,1,0).
- \text{tr\_passive\_in}(1,1,1).
- \text{tr\_passive\_in}(1,1,2).
- \text{tr\_passive\_in}(1,2,0).
- \text{tr\_passive\_in}(1,2,1).
- \text{tr\_passive\_in}(1,2,2).
- \text{tr\_passive\_in}(2,0,0).
- \text{tr\_passive\_in}(2,0,1).
- \text{tr\_passive\_in}(2,0,2).
- \text{tr\_passive\_in}(2,1,0).
- \text{tr\_passive\_in}(2,1,1).
- \text{tr\_passive\_in}(2,1,2).
- \text{tr\_passive\_in}(2,2,0).
- \text{tr\_passive\_in}(2,2,1).
- \text{tr\_passive\_in}(2,2,2).

\textbf{B.4.2} H\_DOLCE\_participation

\textit{H\_PSL\_participates} \neq CQ2 \textbf{Proof attempt for CQ2:}

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce\_participation.p9
(all x all y all t (PC(x,y,t) \rightarrow ED(x) \& PD(y) \& T(t)))).
(all x all t (PD(x) \& PRE(x,t) \rightarrow (exists y PC(y,x,t)))).
(all x (ED(x) \rightarrow (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) \rightarrow PRE(x,t) \& PRE(y,t))).
(all x all y all t (PC(x,y,t) \leftarrow (ED(x) \& PD(y) \& T(t) \& (all t1 ((P(t1,t) \\
& T(t1)) \rightarrow PC(x,y,t1))))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) -> (exists t PRE(x,t)))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t1,t,t2) -> PRE(x,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y)))).
(all x all y (T(x) -> P(x,y))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (T(z) & P(z,x) & -O(z,y)))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  
dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).
(all x (EV(x) | STV(x) -> PD(x))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).
(all x (R(x) -> AB(x))).
(all x (M(x) | F(x) | POB(x) -> PED(x))).
(all x (NPOB(x) -> NPED(x))).
(all x (ACH(x) | ACC(x) -> EV(x))).
(all x (ST(x) | PRO(x) -> STV(x))).
(all x (TL(x) -> TQ(x))).
(all x (SL(x) -> PQ(x))).
(all x (TR(x) | PR(x) | AR(x) -> R(x))).
(all x (APO(x) | NAPO(x) -> POB(x))).
(all x (MOB(x) | SOB(x) -> NPOB(x))).
(all x (T(x) -> TR(x))).
(all x (S(x) -> PR(x))).
(all x (ASO(x) | NASO(x) -> SOB(x))).
(all x (SAG(x) | SC(x) -> ASO(x))).
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).
(all x (ED(x) -> ~PD(x) & ~Q(x) & ~AB(x))).
(all x (PD(x) -> ~Q(x) & ~AB(x))).
(all x (Q(x) -> ~AB(x))).
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x))).
(all x (PED(x) -> ~NPED(x) & ~AS(x))).
(all x (NPED(x) -> ~AS(x))).
(all x (PD(x) <-> EV(x) | STV(x))).
(all x (EV(x) -> ~STV(x))).
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x))).
(all x (TQ(x) -> ~PQ(x) & ~AQ(x))).
(all x (PQ(x) -> ~AQ(x))).
(all x (PED(x) <-> M(x) | F(x) | POB(x))).
(all x (M(x) -> ~F(x) & ~POB(x))).
(all x (F(x) -> ~POB(x))).
(all x (EV(x) <-> ACH(x) | ACC(x))).
(all x (ACH(x) -> ~ACC(x))).
(all x (STV(x) <-> ST(x) | PRO(x))).
(all x (ST(x) -> ~PRO(x))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> ~PR(x) & ~AR(x))).
(all x (PR(x) -> ~AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> ~NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> ~SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> ~NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> ~SC(x))).

%******relevance mappings to requirements
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

% dolce_2_tr
% katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% ’PC equivalent to tr_participates_in’
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

% ’PD equivalent to tr_event’
(all a (PD(a) <-> tr_event(a))).

% ’no dolce_participation equivalent for tr_has_role’

% ’PRE equivalent to tr_exists_at’
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

% ’no dolce_participation equivalent for tr_part_of’

% ’no dolce_participation equivalent for tr_object but we are given the
  additional info that an object is an endurant’) 
(all a (tr_object(a) -> ED(a))).

% ’no dolce_participation equivalent for tr_active_in’

% ’no dolce_participation equivalent for tr_passive_in’

% ’no dolce_participation equivalent for tr_consumed_in’

%******** root theory
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

% gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))
.
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  

dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  

dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/  

dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
\( (\forall x \ (AtP(x) \rightarrow AtP(x))). \)
\( (\forall x \ \forall y \ (U(x,y) \rightarrow U(x,y))). \)

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
\( (\forall x \ (AB(x) \rightarrow AB(x))). \)
\( (\forall x \ (PED(x) \rightarrow PED(x))). \)
\( (\forall x \ (NPED(x) \rightarrow NPED(x))). \)
\( (\forall x \ (AS(x) \rightarrow AS(x))). \)
\( (\forall x \ (EV(x) \rightarrow EV(x))). \)
\( (\forall x \ (STV(x) \rightarrow STV(x))). \)
\( (\forall x \ (TQ(x) \rightarrow TQ(x))). \)
\( (\forall x \ (PQ(x) \rightarrow PQ(x))). \)
\( (\forall x \ (AQ(x) \rightarrow AQ(x))). \)
\( (\forall x \ (R(x) \rightarrow R(x))). \)
\( (\forall x \ (M(x) \rightarrow M(x))). \)
\( (\forall x \ (F(x) \rightarrow F(x))). \)
\( (\forall x \ (POB(x) \rightarrow POB(x))). \)
\( (\forall x \ (NPOB(x) \rightarrow NPOB(x))). \)
\( (\forall x \ (ACH(x) \rightarrow ACH(x))). \)
\( (\forall x \ (ACC(x) \rightarrow ACC(x))). \)
\( (\forall x \ (ST(x) \rightarrow ST(x))). \)
\( (\forall x \ (PRO(x) \rightarrow PRO(x))). \)
\( (\forall x \ (TL(x) \rightarrow TL(x))). \)
\( (\forall x \ (SL(x) \rightarrow SL(x))). \)
\( (\forall x \ (TR(x) \rightarrow TR(x))). \)
\( (\forall x \ (PR(x) \rightarrow PR(x))). \)
\( (\forall x \ (AR(x) \rightarrow AR(x))). \)
\( (\forall x \ (APO(x) \rightarrow APO(x))). \)
\( (\forall x \ (NAPO(x) \rightarrow NAPO(x))). \)
\( (\forall x \ (MOB(x) \rightarrow MOB(x))). \)
\( (\forall x \ (SOB(x) \rightarrow SOB(x))). \)

\( (\forall x \ (S(x) \rightarrow S(x))). \)
\( (\forall x \ (ASO(x) \rightarrow ASO(x))). \)
\( (\forall x \ (NASO(x) \rightarrow NASO(x))). \)
\( (\forall x \ (SAG(x) \rightarrow SAG(x))). \)
\( (\forall x \ (SC(x) \rightarrow SC(x))). \)
\( (\forall x \ (PT(x) \rightarrow PT(x))). \)
\( (\forall x \ (PD(x) \rightarrow PD(x))). \)

\%goal
\%CQ2 'An object must exist in order for it to participate in some event.'
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[-((\forall x \forall a \forall t \ (\text{tr\_participates\_in}(x,a,t) \ \& \ \text{tr\_event}(a) \ \rightarrow \ \text{tr\_exists\_at}(x,t)))\).\]

**Proof found:**

```
-------------------- prooftrans --------------------
Process 4412 was started by meggo on meggo-laptop,
Mon Feb 29 08:26:31 2016
The command was "/cygdrive/c/Program Files (x86)/Prover9-Mace4/bin-win32/prover9".
------------------------ end of head ------------------------

------------------------ end of input ------------------------

------------------------ PROOF ------------------------

% -------- Comments from original proof --------
% Proof 1 at 0.16 (+ 0.03) seconds.
% Length of proof is 12.
% Level of proof is 4.
% Maximum clause weight is 7.
% Given clauses 265.

4 (all x all y all t (PC(x,y,t) \rightarrow \text{PRE}(x,t) \ \& \ \text{PRE}(y,t))) \ # label(non_clause).
    ) . [assumption].
73 (all x all a all t (PC(x,a,t) \leftrightarrow \text{tr\_participates\_in}(x,a,t))) \ # label(
    non_clause). [assumption].
75 (all a all t (\text{PRE}(a,t) \leftrightarrow \text{tr\_exists\_at}(a,t))) \ # label(non_clause). [as
    sumption].
151 -(all x all a all t (\text{tr\_participates\_in}(x,a,t) \ \& \ \text{tr\_event}(a) \ \rightarrow
    \text{tr\_exists\_at}(x,t))) \ # label(non_clause). [assumption].
382 PC(x,y,z) \ | \ -\text{tr\_participates\_in}(x,y,z). [clausify(73)].
384 \text{tr\_participates\_in}(c1,c2,c3). [clausify(151)].
393 -PC(x,y,z) \ | \ \text{PRE}(x,z). [clausify(4)].
426 -\text{PRE}(x,y) \ | \ \text{tr\_exists\_at}(x,y). [clausify(75)].
428 -\text{tr\_exists\_at}(c1,c3). [clausify(151)].
567 PC(c1,c2,c3). [resolve(384,a,382,b)].
726 \text{PRE}(c1,c3). [resolve(567,a,393,a)].
764 $F. [resolve(726,a,426,a),unit\_del(a,428)].
```
HPSL\_participates \not\equiv CQ3  Proof attempt for CQ3:

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) \rightarrow ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) \rightarrow (exists y PC(y,x,t)))).
(all x (ED(x) \rightarrow (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) \rightarrow PRE(x,t) & PRE(y,t))).
(all x all y all t (PC(x,y,t) \leftrightarrow (ED(x) & PD(y) & T(t) & (all t1 ((P(t1,t) & T(t1)) \rightarrow PC(x,y,t1)))).

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) \rightarrow (exists t PRE(x,t)))).
(all x all t all t1 (PRE(x,t) & P(t1,t) \rightarrow PRE(x,t1))).
(all x all t (PRE(x,t) \rightarrow T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t,t1,t2) \rightarrow PRE(x,t))).

\% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all x all y (P(x,y) \rightarrow T(x) & T(y))).
(all x all y (P(x,y) \rightarrow (T(x) \leftrightarrow T(y)))).
(all x all y (T(x) \rightarrow P(x,x))).
(all x all y (T(x) & T(y) & P(x,y) & P(y,x) \rightarrow x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) \rightarrow P(x,z))).
(all x all y (T(x) & T(y) & -P(x,y) \rightarrow (exists z (T(z) & P(z,x) & -O(z,y)))).
(all x all y (T(x) & T(y) & -P(x,y) \rightarrow (exists z (P(z,x) & DJ(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z)))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) \rightarrow y = x)))).
(all x all y (T(x) & T(y) & U(x,y) \rightarrow (exists z (T(z) & (all w (T(w) \rightarrow (O(w,z) <-> O(w,x) | O(w,y))))))).
(all x all y (T(x) & T(y) & O(x,y) \rightarrow (exists z (T(z) & (all w (T(w) \rightarrow (PP(w,z) <-> PP(w,x) & PP(w,y))))))).
\[(\forall x \forall y \forall z (T(x) \land T(y) \land T(z) \rightarrow (\text{SUM}(z,x,y) \leftrightarrow (\forall w (T(w) \rightarrow (O(w,z) \leftrightarrow O(w,x) \lor O(w,y))))))\).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> -PR(x) & -AR(x))).
(all x (PR(x) -> -AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> -NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> -SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> -NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> -SC(x))).

%******relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

%'no dolce_participation equivalent for tr_part_of'

%'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant’)
%(all a (tr_object(a) -> ED(a))).

%'no dolce_participation equivalent for tr_active_in'

%'no dolce_participation equivalent for tr_passive_in'

%'no dolce_participation equivalent for tr_consumed_in'

%**********root theory
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) )))


(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x)));
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t)));
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t)));
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t)));

%gangemi.p9 signature
% _g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x)));
(all x (g_Object(x) -> g_Object(x)));
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y)));
(all x all y (g_pre(x,y) -> g_pre(x,y)));
(all y (g(Time)(y) -> g_Time(y)));

%psl_participation.p9 signature
%_psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x)));
(all x (psl_object(x) -> psl_object(x)));
(all x (psl_timepoint(x) -> psl_timepoint(x)));
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t)));
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t)));
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t)));
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3)));
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2)));
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2)));
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1)));
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)));
(all x psl_endof(x) = psl_endof(x));
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t)));
(all x (ED(x) -> ED(x)));
(all y (PD(y) -> PD(y)));
(all t (T(t) -> T(t)));
(all x all t (PRE(x,t) -> PRE(x,t)));


(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ3 'If something participates in an activity, then this participation translates to any other event that the activity may be a part of.'
-((all x all a_1 all a_2 all t
   (tr_participates_in(x,a_1,t) & tr_part_of(a_1,a_2)
    -> tr_participates_in(x,a_2,t)))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

c1 = 0.
c2 = 1.
c3 = 0.
c4 = 2.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.
psl_endof(2) = 0.
f2(0) = 1.
\[ f_2(1) = 0. \]
\[ f_2(2) = 0. \]

\[ f_3(0) = 2. \]
\[ f_3(1) = 0. \]
\[ f_3(2) = 0. \]

\[ f_5(0) = 2. \]
\[ f_5(1) = 2. \]
\[ f_5(2) = 0. \]

\[ f_{10}(0) = 0. \]
\[ f_{10}(1) = 0. \]
\[ f_{10}(2) = 0. \]

\[ f_1(0,0) = 0. \]
\[ f_1(0,1) = 0. \]
\[ f_1(0,2) = 0. \]
\[ f_1(1,0) = 0. \]
\[ f_1(1,1) = 0. \]
\[ f_1(1,2) = 0. \]
\[ f_1(2,0) = 0. \]
\[ f_1(2,1) = 0. \]
\[ f_1(2,2) = 0. \]

\[ f_6(0,0) = 0. \]
\[ f_6(0,1) = 0. \]
\[ f_6(0,2) = 0. \]
\[ f_6(1,0) = 0. \]
\[ f_6(1,1) = 0. \]
\[ f_6(1,2) = 0. \]
\[ f_6(2,0) = 0. \]
\[ f_6(2,1) = 0. \]
\[ f_6(2,2) = 0. \]

\[ f_7(0,0) = 0. \]
\[ f_7(0,1) = 0. \]
\[ f_7(0,2) = 0. \]
\[ f_7(1,0) = 0. \]
\[ f_7(1,1) = 0. \]
\[ f_7(1,2) = 0. \]
\[ f_7(2,0) = 0. \]
\[ f_7(2,1) = 0. \]
\begin{align*}
\text{f7}(2,2) &= 0. \\
\text{f8}(0,0) &= 0. \\
\text{f8}(0,1) &= 0. \\
\text{f8}(0,2) &= 0. \\
\text{f8}(1,0) &= 0. \\
\text{f8}(1,1) &= 0. \\
\text{f8}(1,2) &= 0. \\
\text{f8}(2,0) &= 0. \\
\text{f8}(2,1) &= 0. \\
\text{f8}(2,2) &= 2. \\
\text{f9}(0,0) &= 0. \\
\text{f9}(0,1) &= 0. \\
\text{f9}(0,2) &= 0. \\
\text{f9}(1,0) &= 0. \\
\text{f9}(1,1) &= 0. \\
\text{f9}(1,2) &= 0. \\
\text{f9}(2,0) &= 0. \\
\text{f9}(2,1) &= 0. \\
\text{f9}(2,2) &= 2. \\
\text{f11}(0,0) &= 0. \\
\text{f11}(0,1) &= 0. \\
\text{f11}(0,2) &= 0. \\
\text{f11}(1,0) &= 0. \\
\text{f11}(1,1) &= 0. \\
\text{f11}(1,2) &= 0. \\
\text{f11}(2,0) &= 0. \\
\text{f11}(2,1) &= 0. \\
\text{f11}(2,2) &= 2. \\
\text{f12}(0,0) &= 0. \\
\text{f12}(0,1) &= 0. \\
\text{f12}(0,2) &= 0. \\
\text{f12}(1,0) &= 0. \\
\text{f12}(1,1) &= 0. \\
\text{f12}(1,2) &= 0. \\
\text{f12}(2,0) &= 0. \\
\text{f12}(2,1) &= 0. \\
\text{f12}(2,2) &= 2. \\
\text{f4}(0,0,0) &= 0. 
\end{align*}
\[ f_4(0,0,1) = 0. \]
\[ f_4(0,0,2) = 0. \]
\[ f_4(0,1,0) = 0. \]
\[ f_4(0,1,1) = 0. \]
\[ f_4(0,1,2) = 0. \]
\[ f_4(0,2,0) = 0. \]
\[ f_4(0,2,1) = 0. \]
\[ f_4(0,2,2) = 0. \]
\[ f_4(1,0,0) = 0. \]
\[ f_4(1,0,1) = 0. \]
\[ f_4(1,0,2) = 0. \]
\[ f_4(1,1,0) = 0. \]
\[ f_4(1,1,1) = 0. \]
\[ f_4(1,1,2) = 0. \]
\[ f_4(1,2,0) = 0. \]
\[ f_4(1,2,1) = 0. \]
\[ f_4(1,2,2) = 0. \]
\[ f_4(2,0,0) = 0. \]
\[ f_4(2,0,1) = 0. \]
\[ f_4(2,0,2) = 0. \]
\[ f_4(2,1,0) = 0. \]
\[ f_4(2,1,1) = 0. \]
\[ f_4(2,1,2) = 0. \]
\[ f_4(2,2,0) = 0. \]
\[ f_4(2,2,1) = 0. \]
\[ f_4(2,2,2) = 0. \]

\[ f_{13}(0,0,0) = 0. \]
\[ f_{13}(0,0,1) = 0. \]
\[ f_{13}(0,0,2) = 0. \]
\[ f_{13}(0,1,0) = 0. \]
\[ f_{13}(0,1,1) = 0. \]
\[ f_{13}(0,1,2) = 0. \]
\[ f_{13}(0,2,0) = 0. \]
\[ f_{13}(0,2,1) = 0. \]
\[ f_{13}(0,2,2) = 0. \]
\[ f_{13}(1,0,0) = 0. \]
\[ f_{13}(1,0,1) = 0. \]
\[ f_{13}(1,0,2) = 0. \]
\[ f_{13}(1,1,0) = 0. \]
\[ f_{13}(1,1,1) = 0. \]
\[ f_{13}(1,1,2) = 0. \]
\[ f_{13}(1,2,0) = 0. \]
f13(1,2,1) = 0.
f13(1,2,2) = 0.
f13(2,0,0) = 0.
f13(2,0,1) = 0.
f13(2,0,2) = 0.
f13(2,1,0) = 0.
f13(2,1,1) = 0.
f13(2,1,2) = 0.
f13(2,2,0) = 0.
f13(2,2,1) = 0.
f13(2,2,2) = 0.

- AB(0).
- AB(1).
  AB(2).

- ACC(0).
- ACC(1).
- ACC(2).

- ACH(0).
- ACH(1).
- ACH(2).

- APO(0).
- APO(1).
- APO(2).

- AQ(0).
- AQ(1).
- AQ(2).

- AR(0).
- AR(1).
- AR(2).

- AS(0).
- AS(1).
- AS(2).

- ASO(0).
- ASO(1).
- ASO(2).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- AtP(0).
- AtP(1).
- AtP(2).

- ED(0).
- ED(1).
- ED(2).

- EV(0).
- EV(1).
- EV(2).

- F(0).
- F(1).
- F(2).

- M(0).
- M(1).
- M(2).

- MOB(0).
- MOB(1).
- MOB(2).

- NAPO(0).
- NAPO(1).
- NAPO(2).

- NASO(0).
- NASO(1).
- NASO(2).

- NPED(0).
- NPED(1).
- NPED(2).

- NPOB(0).
- NPOB(1).
- NPOB(2).

- PD(0).
- PD(1).
- PD(2).
- PED(0).
- PED(1).
- PED(2).

- POB(0).
- POB(1).
- POB(2).

- PQ(0).
- PQ(1).
- PQ(2).

- PR(0).
- PR(1).
- PR(2).

- PRO(0).
- PRO(1).
- PRO(2).

  PT(0).
  PT(1).
  PT(2).

- Q(0).
- Q(1).
- Q(2).

- R(0).
- R(1).
- R(2).

- S(0).
- S(1).
- S(2).

- SAG(0).
- SAG(1).
- SAG(2).

- SC(0).
- SC(1).
- SC(2).

- SL(0).
- SL(1).
- SL(2).

- SOB(0).
- SOB(1).
- SOB(2).

- ST(0).
  ST(1).
  ST(2).

- STV(0).
  STV(1).
  STV(2).

- T(0).
- T(1).
  T(2).

- TL(0).
- TL(1).
- TL(2).

- TQ(0).
- TQ(1).
- TQ(2).

- TR(0).
- TR(1).
  TR(2).

- tr_event(0).
  tr_event(1).
  tr_event(2).

- DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).
- O(0,0).
- O(0,1).
- O(0,2).
- O(1,0).
- O(1,1).
- O(1,2).
- O(2,0).
- O(2,1).
- O(2,2).
- P(0,0).
- P(0,1).
- P(0,2).
- P(1,0).
- P(1,1).
- P(1,2).
- P(2,0).
- P(2,1).
- P(2,2).
- PP(0,0).
- PP(0,1).
- PP(0,2).
- PP(1,0).
- PP(1,1).
- PP(1,2).
- PP(2,0).
- PP(2,1).
- PP(2,2).
- PRE(0,0).
- PRE(0,1).
- PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- PRE(1,2).
- PRE(2,0).
- PRE(2,1).
- PRE(2,2).

- U(0,0).
- U(0,1).
- U(0,2).
- U(1,0).
- U(1,1).
- U(1,2).
- U(2,0).
- U(2,1).
  U(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
  tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
  tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
  tr_exists_at(2,2).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(0,2).
  tr_part_of(1,0).
- tr_part_of(1,1).
  tr_part_of(1,2).
- tr_part_of(2,0).
- tr_part_of(2,1).
  tr_part_of(2,2).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,0,2).
- PC(0,1,0).
- PC(0,1,1).
  PC(0,1,2).
- PC(0,2,0).
- PC(0,2,1).
- PC(0,2,2).
- PC(1,0,0).
- \texttt{PC(1,0,1)}.
- \texttt{PC(1,0,2)}.
- \texttt{PC(1,1,0)}.
- \texttt{PC(1,1,1)}.
- \texttt{PC(1,1,2)}.
- \texttt{PC(1,2,0)}.
- \texttt{PC(1,2,1)}.
- \texttt{PC(1,2,2)}.
- \texttt{PC(2,0,0)}.
- \texttt{PC(2,0,1)}.
- \texttt{PC(2,0,2)}.
- \texttt{PC(2,1,0)}.
- \texttt{PC(2,1,1)}.
- \texttt{PC(2,1,2)}.
- \texttt{PC(2,2,0)}.
- \texttt{PC(2,2,1)}.
- \texttt{PC(2,2,2)}.
- \texttt{SUM(0,0,0)}.
- \texttt{SUM(0,0,1)}.
- \texttt{SUM(0,0,2)}.
- \texttt{SUM(0,1,0)}.
- \texttt{SUM(0,1,1)}.
- \texttt{SUM(0,1,2)}.
- \texttt{SUM(0,2,0)}.
- \texttt{SUM(0,2,1)}.
- \texttt{SUM(0,2,2)}.
- \texttt{SUM(1,0,0)}.
- \texttt{SUM(1,0,1)}.
- \texttt{SUM(1,0,2)}.
- \texttt{SUM(1,1,0)}.
- \texttt{SUM(1,1,1)}.
- \texttt{SUM(1,1,2)}.
- \texttt{SUM(1,2,0)}.
- \texttt{SUM(1,2,1)}.
- \texttt{SUM(1,2,2)}.
- \texttt{SUM(2,0,0)}.
- \texttt{SUM(2,0,1)}.
- \texttt{SUM(2,0,2)}.
- \texttt{SUM(2,1,0)}.
- \texttt{SUM(2,1,1)}.
- \texttt{SUM(2,1,2)}.
- \texttt{SUM(2,2,0)}.
\textbf{APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES} 412

- \text{SUM}(2,2,1).
- \text{SUM}(2,2,2).

- \text{tr} \text{\_participates\_in}(0,0,0).
- \text{tr} \text{\_participates\_in}(0,0,1).
- \text{tr} \text{\_participates\_in}(0,0,2).
- \text{tr} \text{\_participates\_in}(0,1,0).
- \text{tr} \text{\_participates\_in}(0,1,1).
- \text{tr} \text{\_participates\_in}(0,1,2).
- \text{tr} \text{\_participates\_in}(0,2,0).
- \text{tr} \text{\_participates\_in}(0,2,1).
- \text{tr} \text{\_participates\_in}(0,2,2).
- \text{tr} \text{\_participates\_in}(1,0,0).
- \text{tr} \text{\_participates\_in}(1,0,1).
- \text{tr} \text{\_participates\_in}(1,0,2).
- \text{tr} \text{\_participates\_in}(1,1,0).
- \text{tr} \text{\_participates\_in}(1,1,1).
- \text{tr} \text{\_participates\_in}(1,1,2).
- \text{tr} \text{\_participates\_in}(1,2,0).
- \text{tr} \text{\_participates\_in}(1,2,1).
- \text{tr} \text{\_participates\_in}(1,2,2).
- \text{tr} \text{\_participates\_in}(2,0,0).
- \text{tr} \text{\_participates\_in}(2,0,1).
- \text{tr} \text{\_participates\_in}(2,0,2).
- \text{tr} \text{\_participates\_in}(2,1,0).
- \text{tr} \text{\_participates\_in}(2,1,1).
- \text{tr} \text{\_participates\_in}(2,1,2).
- \text{tr} \text{\_participates\_in}(2,2,0).
- \text{tr} \text{\_participates\_in}(2,2,1).
- \text{tr} \text{\_participates\_in}(2,2,2).

\textit{\text{HPSL\_participates}} \not\models \text{CQ4} \text{ Proof attempt for CQ4:}

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x,t) -> (exists y PC(y,x,t)))).
(all x (ED(x) -> (exists y exists t PC(x,y,t)))).
(all x all y all t (PC(x,y,t) -> PRE(x,t) & PRE(y,t))).
(all x all y all t (PC(x,y,t) <-> (ED(x) & PD(y) & T(t) & (all t1 ((P(t1,t)
  & T(t1)) -> PC(x,y,t1)))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x)))).
(all x all t all t1 (PRE(x,t) & P(t1,t) -> PRE(x,t1))).
(all x all t (PRE(x,t) -> T(t))).
(all x all t all t1 all t2 (PRE(x,t1) & PRE(x,t2) & SUM(t1,t2,t) -> PRE(x,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all x all y (P(x,y) -> T(x) & T(y))).
(all x all y (P(x,y) -> (T(x) <-> T(y)))).
(all x all y (T(x) -> P(x,y))).
(all x all y (T(x) & P(x,y) & P(y,x) -> x = y)).
(all x all y all z (T(x) & T(y) & P(x,y) & P(y,z) -> P(x,z))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (T(z) & P(z,x) & -O(z,y)))).
(all x all y (T(x) & T(y) & -P(x,y) -> (exists z (P(z,x) & DJ(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (PP(x,y) <-> P(x,y) & -P(y,x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)))).
(all x all y (T(x) & T(y) -> (DJ(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(z,x) & P(y,z) & T(z)))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (PP(w,z) <-> PP(w,x) & PP(w,y)))))])).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y)))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x)))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x)))).
(all x (EV(x) | STV(x) -> PD(x)))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x)))).
(all x (R(x) -> AB(x)))).
(all x (M(x) | F(x) | POB(x) -> PED(x)))).
(all x (NFOB(x) -> NPED(x)))).
(all x (ACH(x) | ACC(x) -> EV(x)))).
(all x (ST(x) | PRO(x) -> STV(x)))).
(all x (TL(x) -> TQ(x)))
(all x (SL(x) -> PQ(x)))
(all x (TR(x) | PR(x) | AR(x) -> R(x)))
(all x (APO(x) | NAPO(x) -> POB(x)))
(all x (MOB(x) | SOB(x) -> NPOB(x)))
(all x (T(x) -> TR(x)))
(all x (S(x) -> PR(x)))
(all x (ASO(x) | NASO(x) -> SOB(x)))
(all x (SAG(x) | SC(x) -> ASO(x)))
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x)))
(all x (ED(x) -> -PD(x) & ~Q(x) & ~AB(x)))
(all x (PD(x) -> ~Q(x) & ~AB(x)))
(all x (Q(x) -> ~AB(x)))
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x)))
(all x (PED(x) -> ~NPED(x) & ~AS(x)))
(all x (NPED(x) -> ~AS(x)))
(all x (PD(x) <-> EV(x) | STV(x)))
(all x (EV(x) -> ~STV(x)))
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x)))
(all x (TQ(x) -> ~PQ(x) & ~AQ(x)))
(all x (PQ(x) -> ~AQ(x)))
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x)))
(all x (TQ(x) -> ~PQ(x) & ~AQ(x)))
(all x (AQ(x) -> ~AQ(x)))
(all x (PED(x) <-> M(x) | F(x) | POB(x)))
(all x (M(x) -> ~F(x) & ~POB(x)))
(all x (F(x) -> ~POB(x)))
(all x (EV(x) <-> ACH(x) | ACC(x)))
(all x (ACH(x) -> ~ACC(x)))
(all x (STV(x) <-> ST(x) | PRO(x)))
(all x (STV(x) <-> ST(x) | PRO(x)))
(all x (R(x) <-> TR(x) | PR(x) | AR(x)))
(all x (TR(x) -> ~PR(x) & ~AR(x)))
(all x (PR(x) -> ~AR(x)))
(all x (POB(x) <-> APO(x) | NAPO(x)))
(all x (APO(x) -> ~NAPO(x)))
(all x (NPOB(x) <-> MOB(x) | SOB(x)))
(all x (MOB(x) -> ~SOB(x)))
(all x (SOB(x) <-> ASO(x) | NASO(x)))
(all x (ASO(x) -> ~NASO(x)))
(all x (ASO(x) <-> SAG(x) | SC(x)))
(all x (SAG(x) -> ~SC(x)))

***** relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

% 'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

% 'no dolce_participation equivalent for tr_has_role'

% 'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

% 'no dolce_participation equivalent for tr_part_of'

% 'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant'
%(all a (tr_object(a) -> ED(a))).

% 'no dolce_participation equivalent for tr_active_in'

% 'no dolce_participation equivalent for tr_passive_in'

% 'no dolce_participation equivalent for tr_consumed_in'

%**********root theory
%Root Theory (collection of candidates' signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) ))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all x all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
%Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ4 'Objects can not participate in more than one different activity
-((all x all a_1 all a_2
   (tr_object(x)
\& \text{tr\_participates\_in}(x, a_1, t) \\
\& \text{tr\_participates\_in}(x, a_2, t) \\
\rightarrow \\
((a_1 = a_2) \mid \text{tr\_part\_of}(a_1, a_2))).

\textbf{Counterexample found:}

\%
\% number = 1
\%
\% seconds = 0

\%
\% Interpretation of size 4

\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\% 
\%
\%
\[ f_{10}(0) = 0. \]
\[ f_{10}(1) = 0. \]
\[ f_{10}(2) = 0. \]
\[ f_{10}(3) = 0. \]

\[ f_1(0,0) = 0. \]
\[ f_1(0,1) = 0. \]
\[ f_1(0,2) = 0. \]
\[ f_1(0,3) = 0. \]
\[ f_1(1,0) = 0. \]
\[ f_1(1,1) = 0. \]
\[ f_1(1,2) = 0. \]
\[ f_1(1,3) = 0. \]
\[ f_1(2,0) = 1. \]
\[ f_1(2,1) = 0. \]
\[ f_1(2,2) = 0. \]
\[ f_1(2,3) = 0. \]
\[ f_1(3,0) = 1. \]
\[ f_1(3,1) = 0. \]
\[ f_1(3,2) = 0. \]
\[ f_1(3,3) = 0. \]

\[ f_6(0,0) = 0. \]
\[ f_6(0,1) = 0. \]
\[ f_6(0,2) = 0. \]
\[ f_6(0,3) = 0. \]
\[ f_6(1,0) = 0. \]
\[ f_6(1,1) = 0. \]
\[ f_6(1,2) = 0. \]
\[ f_6(1,3) = 0. \]
\[ f_6(2,0) = 0. \]
\[ f_6(2,1) = 0. \]
\[ f_6(2,2) = 0. \]
\[ f_6(2,3) = 0. \]
\[ f_6(3,0) = 0. \]
\[ f_6(3,1) = 0. \]
\[ f_6(3,2) = 0. \]
\[ f_6(3,3) = 0. \]

\[ f_7(0,0) = 0. \]
\[ f_7(0,1) = 0. \]
\[ f_7(0,2) = 0. \]
\begin{align*}
f_7(0,3) &= 0. \\
f_7(1,0) &= 0. \\
f_7(1,1) &= 0. \\
f_7(1,2) &= 0. \\
f_7(1,3) &= 0. \\
f_7(2,0) &= 0. \\
f_7(2,1) &= 0. \\
f_7(2,2) &= 0. \\
f_7(2,3) &= 0. \\
f_7(3,0) &= 0. \\
f_7(3,1) &= 0. \\
f_7(3,2) &= 0. \\
f_7(3,3) &= 0. \\
\end{align*}

\begin{align*}
f_8(0,0) &= 0. \\
f_8(0,1) &= 0. \\
f_8(0,2) &= 0. \\
f_8(0,3) &= 0. \\
f_8(1,0) &= 0. \\
f_8(1,1) &= 0. \\
f_8(1,2) &= 0. \\
f_8(1,3) &= 0. \\
f_8(2,0) &= 0. \\
f_8(2,1) &= 0. \\
f_8(2,2) &= 0. \\
f_8(2,3) &= 0. \\
f_8(3,0) &= 0. \\
f_8(3,1) &= 0. \\
f_8(3,2) &= 0. \\
f_8(3,3) &= 0. \\
\end{align*}

\begin{align*}
f_9(0,0) &= 0. \\
f_9(0,1) &= 0. \\
f_9(0,2) &= 0. \\
f_9(0,3) &= 0. \\
f_9(1,0) &= 0. \\
f_9(1,1) &= 0. \\
f_9(1,2) &= 0. \\
f_9(1,3) &= 0. \\
f_9(2,0) &= 0. \\
f_9(2,1) &= 0. \\
f_9(2,2) &= 0. \\
f_9(2,3) &= 0. \\
\end{align*}
\textbf{APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES}

\begin{verbatim}
f9(3,0) = 0.
f9(3,1) = 0.
f9(3,2) = 0.
f9(3,3) = 0.

f11(0,0) = 0.
f11(0,1) = 0.
f11(0,2) = 0.
f11(0,3) = 0.
f11(1,0) = 0.
f11(1,1) = 0.
f11(1,2) = 0.
f11(1,3) = 0.
f11(2,0) = 0.
f11(2,1) = 0.
f11(2,2) = 0.
f11(2,3) = 0.
f11(3,0) = 0.
f11(3,1) = 0.
f11(3,2) = 0.
f11(3,3) = 0.

f12(0,0) = 0.
f12(0,1) = 0.
f12(0,2) = 0.
f12(0,3) = 0.
f12(1,0) = 0.
f12(1,1) = 0.
f12(1,2) = 0.
f12(1,3) = 0.
f12(2,0) = 0.
f12(2,1) = 0.
f12(2,2) = 0.
f12(2,3) = 0.
f12(3,0) = 0.
f12(3,1) = 0.
f12(3,2) = 0.
f12(3,3) = 0.

f4(0,0,0) = 0.
f4(0,0,1) = 0.
f4(0,0,2) = 0.
f4(0,0,3) = 0.
\end{verbatim}
\[ f_4(0,1,0) = 0. \]
\[ f_4(0,1,1) = 0. \]
\[ f_4(0,1,2) = 0. \]
\[ f_4(0,1,3) = 0. \]
\[ f_4(0,2,0) = 0. \]
\[ f_4(0,2,1) = 0. \]
\[ f_4(0,2,2) = 0. \]
\[ f_4(0,2,3) = 0. \]
\[ f_4(0,3,0) = 0. \]
\[ f_4(0,3,1) = 0. \]
\[ f_4(0,3,2) = 0. \]
\[ f_4(0,3,3) = 0. \]
\[ f_4(1,0,0) = 0. \]
\[ f_4(1,0,1) = 0. \]
\[ f_4(1,0,2) = 0. \]
\[ f_4(1,0,3) = 0. \]
\[ f_4(1,1,0) = 0. \]
\[ f_4(1,1,1) = 0. \]
\[ f_4(1,1,2) = 0. \]
\[ f_4(1,1,3) = 0. \]
\[ f_4(1,2,0) = 0. \]
\[ f_4(1,2,1) = 0. \]
\[ f_4(1,2,2) = 0. \]
\[ f_4(1,2,3) = 0. \]
\[ f_4(1,3,0) = 0. \]
\[ f_4(1,3,1) = 0. \]
\[ f_4(1,3,2) = 0. \]
\[ f_4(1,3,3) = 0. \]
\[ f_4(2,0,0) = 0. \]
\[ f_4(2,0,1) = 0. \]
\[ f_4(2,0,2) = 0. \]
\[ f_4(2,0,3) = 0. \]
\[ f_4(2,1,0) = 0. \]
\[ f_4(2,1,1) = 0. \]
\[ f_4(2,1,2) = 0. \]
\[ f_4(2,1,3) = 0. \]
\[ f_4(2,2,0) = 0. \]
\[ f_4(2,2,1) = 0. \]
\[ f_4(2,2,2) = 0. \]
\[ f_4(2,2,3) = 0. \]
\[ f_4(2,3,0) = 0. \]
\[ f_4(2,3,1) = 0. \]
\[ f_4(2,3,2) = 0. \]
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[ f_4(2, 3, 3) = 0. \]
\[ f_4(3, 0, 0) = 0. \]
\[ f_4(3, 0, 1) = 0. \]
\[ f_4(3, 0, 2) = 0. \]
\[ f_4(3, 0, 3) = 0. \]
\[ f_4(3, 1, 0) = 0. \]
\[ f_4(3, 1, 1) = 0. \]
\[ f_4(3, 1, 2) = 0. \]
\[ f_4(3, 1, 3) = 0. \]
\[ f_4(3, 2, 0) = 0. \]
\[ f_4(3, 2, 1) = 0. \]
\[ f_4(3, 2, 2) = 0. \]
\[ f_4(3, 2, 3) = 0. \]
\[ f_4(3, 3, 0) = 0. \]
\[ f_4(3, 3, 1) = 0. \]
\[ f_4(3, 3, 2) = 0. \]
\[ f_4(3, 3, 3) = 0. \]

\[ f_{13}(0, 0, 0) = 0. \]
\[ f_{13}(0, 0, 1) = 0. \]
\[ f_{13}(0, 0, 2) = 0. \]
\[ f_{13}(0, 0, 3) = 0. \]
\[ f_{13}(0, 1, 0) = 0. \]
\[ f_{13}(0, 1, 1) = 0. \]
\[ f_{13}(0, 1, 2) = 0. \]
\[ f_{13}(0, 1, 3) = 0. \]
\[ f_{13}(0, 2, 0) = 0. \]
\[ f_{13}(0, 2, 1) = 0. \]
\[ f_{13}(0, 2, 2) = 0. \]
\[ f_{13}(0, 2, 3) = 0. \]
\[ f_{13}(0, 3, 0) = 0. \]
\[ f_{13}(0, 3, 1) = 0. \]
\[ f_{13}(0, 3, 2) = 0. \]
\[ f_{13}(0, 3, 3) = 0. \]
\[ f_{13}(1, 0, 0) = 0. \]
\[ f_{13}(1, 0, 1) = 0. \]
\[ f_{13}(1, 0, 2) = 0. \]
\[ f_{13}(1, 0, 3) = 0. \]
\[ f_{13}(1, 1, 0) = 0. \]
\[ f_{13}(1, 1, 1) = 0. \]
\[ f_{13}(1, 1, 2) = 0. \]
\[ f_{13}(1, 1, 3) = 0. \]
\[ f_{13}(1, 2, 0) = 0. \]
\[ f_{13}(1,2,1) = 0. \]
\[ f_{13}(1,2,2) = 0. \]
\[ f_{13}(1,2,3) = 0. \]
\[ f_{13}(1,3,0) = 0. \]
\[ f_{13}(1,3,1) = 0. \]
\[ f_{13}(1,3,2) = 0. \]
\[ f_{13}(1,3,3) = 0. \]
\[ f_{13}(2,0,0) = 0. \]
\[ f_{13}(2,0,1) = 0. \]
\[ f_{13}(2,0,2) = 0. \]
\[ f_{13}(2,0,3) = 0. \]
\[ f_{13}(2,1,0) = 0. \]
\[ f_{13}(2,1,1) = 0. \]
\[ f_{13}(2,1,2) = 0. \]
\[ f_{13}(2,1,3) = 0. \]
\[ f_{13}(2,2,0) = 0. \]
\[ f_{13}(2,2,1) = 0. \]
\[ f_{13}(2,2,2) = 0. \]
\[ f_{13}(2,2,3) = 0. \]
\[ f_{13}(2,3,0) = 0. \]
\[ f_{13}(2,3,1) = 0. \]
\[ f_{13}(2,3,2) = 0. \]
\[ f_{13}(2,3,3) = 0. \]
\[ f_{13}(3,0,0) = 0. \]
\[ f_{13}(3,0,1) = 0. \]
\[ f_{13}(3,0,2) = 0. \]
\[ f_{13}(3,0,3) = 0. \]
\[ f_{13}(3,1,0) = 0. \]
\[ f_{13}(3,1,1) = 0. \]
\[ f_{13}(3,1,2) = 0. \]
\[ f_{13}(3,1,3) = 0. \]
\[ f_{13}(3,2,0) = 0. \]
\[ f_{13}(3,2,1) = 0. \]
\[ f_{13}(3,2,2) = 0. \]
\[ f_{13}(3,2,3) = 0. \]
\[ f_{13}(3,3,0) = 0. \]
\[ f_{13}(3,3,1) = 0. \]
\[ f_{13}(3,3,2) = 0. \]
\[ f_{13}(3,3,3) = 0. \]

\[ AB(0). \]
- \[ AB(1). \]
- \[ AB(2). \]
- AB(3).
- ACC(0).
- ACC(1).
- ACC(2).
- ACC(3).
- ACH(0).
- ACH(1).
- ACH(2).
- ACH(3).
- APO(0).
- APO(1).
- APO(2).
- APO(3).
- AQ(0).
- AQ(1).
- AQ(2).
- AQ(3).
- AR(0).
- AR(1).
- AR(2).
- AR(3).
- AS(0).
- AS(1).
- AS(2).
- AS(3).
- ASO(0).
- ASO(1).
- ASO(2).
- ASO(3).

AtP(0).
- AtP(1).
- AtP(2).
- AtP(3).
- ED(0).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- ED(1).
- ED(2).
- ED(3).
- EV(0).
- EV(1).
- EV(2).
- EV(3).
- F(0).
- F(1).
- F(2).
- F(3).
- M(0).
- M(1).
- M(2).
- M(3).
- MOB(0).
- MOB(1).
- MOB(2).
- MOB(3).
- NAPO(0).
- NAPO(1).
- NAPO(2).
- NAPO(3).
- NASO(0).
- NASO(1).
- NASO(2).
- NASO(3).
- NPED(0).
  - NPED(1).
  - NPED(2).
  - NPED(3).
- NPOB(0).
- NPOB(1).
- NPOB(2).
- NPOB(3).
Appendix B. Comparing Participation Ontologies: Proof Files

- PD(0).
- PD(1).
  PD(2).
  PD(3).

- PED(0).
- PED(1).
- PED(2).
- PED(3).

- POB(0).
- POB(1).
- POB(2).
- POB(3).

- PQ(0).
- PQ(1).
- PQ(2).
- PQ(3).

- PR(0).
- PR(1).
- PR(2).
- PR(3).

- PRO(0).
- PRO(1).
- PRO(2).
- PRO(3).

PT(0).
PT(1).
PT(2).
PT(3).

- Q(0).
- Q(1).
- Q(2).
- Q(3).

R(0).
R(1).
- R(2).
- R(3).

- S(0).
- S(1).
- S(2).
- S(3).

- SAG(0).
- SAG(1).
- SAG(2).
- SAG(3).

- SC(0).
- SC(1).
- SC(2).
- SC(3).

- SL(0).
- SL(1).
- SL(2).
- SL(3).

- SOB(0).
- SOB(1).
- SOB(2).
- SOB(3).

- ST(0).
- ST(1).
- ST(2).
- ST(3).

- STV(0).
- STV(1).
- STV(2).
- STV(3).

T(0).
- T(1).
- T(2).
- T(3).
- TL(0).
- TL(1).
- TL(2).
- TL(3).

- TQ(0).
- TQ(1).
- TQ(2).
- TQ(3).

- TR(0).
- TR(1).
- TR(2).
- TR(3).

- tr_event(0).
  - tr_event(1).
    - tr_event(2).
    - tr_event(3).

- tr_object(0).
  - tr_object(1).
  - tr_object(2).
  - tr_object(3).

- DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(0,3).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(1,3).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).
- DJ(2,3).
- DJ(3,0).
- DJ(3,1).
- DJ(3,2).
- DJ(3,3).

0(0,0).
- \( O(0,1) \).
- \( O(0,2) \).
- \( O(0,3) \).
- \( O(1,0) \).
- \( O(1,1) \).
- \( O(1,2) \).
- \( O(1,3) \).
- \( O(2,0) \).
- \( O(2,1) \).
- \( O(2,2) \).
- \( O(2,3) \).
- \( O(3,0) \).
- \( O(3,1) \).
- \( O(3,2) \).
- \( O(3,3) \).

- \( P(0,0) \).
- \( P(0,1) \).
- \( P(0,2) \).
- \( P(0,3) \).
- \( P(1,0) \).
- \( P(1,1) \).
- \( P(1,2) \).
- \( P(1,3) \).
- \( P(2,0) \).
- \( P(2,1) \).
- \( P(2,2) \).
- \( P(2,3) \).
- \( P(3,0) \).
- \( P(3,1) \).
- \( P(3,2) \).
- \( P(3,3) \).

- \( PP(0,0) \).
- \( PP(0,1) \).
- \( PP(0,2) \).
- \( PP(0,3) \).
- \( PP(1,0) \).
- \( PP(1,1) \).
- \( PP(1,2) \).
- \( PP(1,3) \).
- \( PP(2,0) \).
- \( PP(2,1) \).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- PP(2,2).
- PP(2,3).
- PP(3,0).
- PP(3,1).
- PP(3,2).
- PP(3,3).
- PRE(0,0).
- PRE(0,1).
- PRE(0,2).
- PRE(0,3).
  - PRE(1,0).
  - PRE(1,1).
  - PRE(1,2).
  - PRE(1,3).
  - PRE(2,0).
  - PRE(2,1).
  - PRE(2,2).
  - PRE(2,3).
  - PRE(3,0).
  - PRE(3,1).
  - PRE(3,2).
  - PRE(3,3).

  U(0,0).
  - U(0,1).
  - U(0,2).
  - U(0,3).
  - U(1,0).
  - U(1,1).
  - U(1,2).
  - U(1,3).
  - U(2,0).
  - U(2,1).
  - U(2,2).
  - U(2,3).
  - U(3,0).
  - U(3,1).
  - U(3,2).
  - U(3,3).

  - tr_exists_at(0,0).
  - tr_exists_at(0,1).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

- tr_exists_at(0, 2).
- tr_exists_at(0, 3).
  tr_exists_at(1, 0).
- tr_exists_at(1, 1).
- tr_exists_at(1, 2).
- tr_exists_at(1, 3).
  tr_exists_at(2, 0).
- tr_exists_at(2, 1).
- tr_exists_at(2, 2).
- tr_exists_at(2, 3).
  tr_exists_at(3, 0).
- tr_exists_at(3, 1).
- tr_exists_at(3, 2).
- tr_exists_at(3, 3).

- tr_part_of(0, 0).
- tr_part_of(0, 1).
- tr_part_of(0, 2).
- tr_part_of(0, 3).
- tr_part_of(1, 0).
- tr_part_of(1, 1).
- tr_part_of(1, 2).
- tr_part_of(1, 3).
- tr_part_of(2, 0).
- tr_part_of(2, 1).
- tr_part_of(2, 2).
- tr_part_of(2, 3).
- tr_part_of(3, 0).
- tr_part_of(3, 1).
- tr_part_of(3, 2).
- tr_part_of(3, 3).

- PC(0, 0, 0).
- PC(0, 0, 1).
- PC(0, 0, 2).
- PC(0, 0, 3).
- PC(0, 1, 0).
- PC(0, 1, 1).
- PC(0, 1, 2).
- PC(0, 1, 3).
- PC(0, 2, 0).
- PC(0, 2, 1).
- PC(0, 2, 2).
- PC(0, 2, 3).
- PC(0, 3, 0).
- PC(0, 3, 1).
- PC(0, 3, 2).
- PC(0, 3, 3).
- PC(1, 0, 0).
- PC(1, 0, 1).
- PC(1, 0, 2).
- PC(1, 0, 3).
- PC(1, 1, 0).
- PC(1, 1, 1).
- PC(1, 1, 2).
- PC(1, 1, 3).
- PC(1, 2, 0).
- PC(1, 2, 1).
- PC(1, 2, 2).
- PC(1, 2, 3).
- PC(1, 3, 0).
- PC(1, 3, 1).
- PC(1, 3, 2).
- PC(1, 3, 3).
- PC(2, 0, 0).
- PC(2, 0, 1).
- PC(2, 0, 2).
- PC(2, 0, 3).
- PC(2, 1, 0).
- PC(2, 1, 1).
- PC(2, 1, 2).
- PC(2, 1, 3).
- PC(2, 2, 0).
- PC(2, 2, 1).
- PC(2, 2, 2).
- PC(2, 2, 3).
- PC(2, 3, 0).
- PC(2, 3, 1).
- PC(2, 3, 2).
- PC(2, 3, 3).
- PC(3, 0, 0).
- PC(3, 0, 1).
- PC(3, 0, 2).
- PC(3, 0, 3).
- PC(3, 1, 0).
- PC(3, 1, 1).
- PC(3,1,2).
- PC(3,1,3).
- PC(3,2,0).
- PC(3,2,1).
- PC(3,2,2).
- PC(3,2,3).
- PC(3,3,0).
- PC(3,3,1).
- PC(3,3,2).
- PC(3,3,3).

  SUM(0,0,0).
- SUM(0,0,1).
- SUM(0,0,2).
- SUM(0,0,3).
- SUM(0,1,0).
- SUM(0,1,1).
- SUM(0,1,2).
- SUM(0,1,3).
- SUM(0,2,0).
- SUM(0,2,1).
- SUM(0,2,2).
- SUM(0,2,3).
- SUM(0,3,0).
- SUM(0,3,1).
- SUM(0,3,2).
- SUM(0,3,3).
- SUM(1,0,0).
- SUM(1,0,1).
- SUM(1,0,2).
- SUM(1,0,3).
- SUM(1,1,0).
- SUM(1,1,1).
- SUM(1,1,2).
- SUM(1,1,3).
- SUM(1,2,0).
- SUM(1,2,1).
- SUM(1,2,2).
- SUM(1,2,3).
- SUM(1,3,0).
- SUM(1,3,1).
- SUM(1,3,2).
- SUM(1,3,3).
- \text{SUM}(2,0,0).
- \text{SUM}(2,0,1).
- \text{SUM}(2,0,2).
- \text{SUM}(2,0,3).
- \text{SUM}(2,1,0).
- \text{SUM}(2,1,1).
- \text{SUM}(2,1,2).
- \text{SUM}(2,1,3).
- \text{SUM}(2,2,0).
- \text{SUM}(2,2,1).
- \text{SUM}(2,2,2).
- \text{SUM}(2,2,3).
- \text{SUM}(2,3,0).
- \text{SUM}(2,3,1).
- \text{SUM}(2,3,2).
- \text{SUM}(2,3,3).
- \text{SUM}(3,0,0).
- \text{SUM}(3,0,1).
- \text{SUM}(3,0,2).
- \text{SUM}(3,0,3).
- \text{SUM}(3,1,0).
- \text{SUM}(3,1,1).
- \text{SUM}(3,1,2).
- \text{SUM}(3,1,3).
- \text{SUM}(3,2,0).
- \text{SUM}(3,2,1).
- \text{SUM}(3,2,2).
- \text{SUM}(3,2,3).
- \text{SUM}(3,3,0).
- \text{SUM}(3,3,1).
- \text{SUM}(3,3,2).
- \text{SUM}(3,3,3).

- \text{tr_participates_in}(0,0,0).
- \text{tr_participates_in}(0,0,1).
- \text{tr_participates_in}(0,0,2).
- \text{tr_participates_in}(0,0,3).
- \text{tr_participates_in}(0,1,0).
- \text{tr_participates_in}(0,1,1).
- \text{tr_participates_in}(0,1,2).
- \text{tr_participates_in}(0,1,3).
- \text{tr_participates_in}(0,2,0).
- \text{tr_participates_in}(0,2,1).
- tr_participates_in(0,2,2).
- tr_participates_in(0,2,3).
- tr_participates_in(0,3,0).
- tr_participates_in(0,3,1).
- tr_participates_in(0,3,2).
- tr_participates_in(0,3,3).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,0,2).
- tr_participates_in(1,0,3).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).
- tr_participates_in(1,1,2).
- tr_participates_in(1,1,3).
  tr_participates_in(1,2,0).
- tr_participates_in(1,2,1).
- tr_participates_in(1,2,2).
- tr_participates_in(1,2,3).
  tr_participates_in(1,3,0).
- tr_participates_in(1,3,1).
- tr_participates_in(1,3,2).
- tr_participates_in(1,3,3).
- tr_participates_in(2,0,0).
- tr_participates_in(2,0,1).
- tr_participates_in(2,0,2).
- tr_participates_in(2,0,3).
- tr_participates_in(2,1,0).
- tr_participates_in(2,1,1).
- tr_participates_in(2,1,2).
- tr_participates_in(2,1,3).
- tr_participates_in(2,2,0).
- tr_participates_in(2,2,1).
- tr_participates_in(2,2,2).
- tr_participates_in(2,2,3).
- tr_participates_in(2,3,0).
- tr_participates_in(2,3,1).
- tr_participates_in(2,3,2).
- tr_participates_in(2,3,3).
- tr_participates_in(3,0,0).
- tr_participates_in(3,0,1).
- tr_participates_in(3,0,2).
- tr_participates_in(3,0,3).
- tr_participates_in(3,1,0).
- tr_participates_in(3,1,1).
- tr_participates_in(3,1,2).
- tr_participates_in(3,1,3).
- tr_participates_in(3,2,0).
- tr_participates_in(3,2,1).
- tr_participates_in(3,2,2).
- tr_participates_in(3,2,3).
- tr_participates_in(3,3,0).
- tr_participates_in(3,3,1).
- tr_participates_in(3,3,2).
- tr_participates_in(3,3,3).

\[ H_{\text{PSL_participates}} \neq CQ5 \]

**Proof attempt for CQ5:**

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x, y, t) \rightarrow ED(x) & PD(y) & T(t))).
(all x all t (PD(x) & PRE(x, t) \rightarrow \exists y PC(y, x, t))).
(all x (ED(x) \rightarrow \exists y \exists t PC(x, y, t))).
(all x all y all t (PC(x, y, t) \rightarrow PRE(x, t) & PRE(y, t))).
(all x all y all t (PC(x, y, t) \leftrightarrow (ED(x) & PD(y) & T(t)) \& (all t1 ((P(t1, t)
\& T(t1)) \rightarrow PC(x, y, t1))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_present.p9
(all x (ED(x) | PD(x) | Q(x) \rightarrow \exists t PRE(x, t))).
(all x all t all t1 (PRE(x, t) \& P(t1, t) \rightarrow PRE(x, t1))).
(all x all t (PRE(x, t) \rightarrow T(t))).
(all x all t all t1 all t2 (PRE(x, t1) \& PRE(x, t2) \& SUM(t, t1, t2) \rightarrow PRE(x, t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_time_mereology.p9
(all x all y (P(x, y) \rightarrow T(x) \& T(y))).
(all x all y (P(x, y) \rightarrow (T(x) \leftrightarrow T(y)))).
(all x all y (T(x) \rightarrow P(x, x))).
(all x all y (T(x) & T(y) & P(x, y) & P(y, x) \rightarrow x = y)).
(all x all y all z (T(x) & T(y) & P(x, y) & P(y, z) \rightarrow P(x, z))).
(all x all y (T(x) & T(y) & \neg P(x, y) \rightarrow (\exists z (T(z) & P(z, x) & \neg O(z, y)))).
(all x all y (T(x) & T(y) & \neg P(x, y) \rightarrow (\exists z (P(z, x) & DJ(z, y) & T(z)))).
(all x all y (T(x) & T(y) \rightarrow (PP(x, y) \leftrightarrow P(x, y) \& \neg P(y, x)))).
(all x all y (T(x) & T(y) -> (O(x,y) <-> (exists z (P(z,x) & P(z,y) & T(z)) )))).
(all x all y (T(x) & T(y) -> (D(x,y) <-> -O(x,y)))).
(all x all y (T(x) & T(y) -> (U(x,y) <-> (exists z (P(x,z) & P(y,z) & T(z)) )))).
(all x (AtP(x) <-> T(x) & (all y (T(y) & P(y,x) -> y = x)))).
(all x all y (T(x) & T(y) & U(x,y) -> (exists z (T(z) & (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y)))))))).
(all x all y (T(x) & T(y) & O(x,y) -> (exists z (T(z) & (all w (T(w) -> (P(w,z) <-> P(w,x) & P(w,y)))))))).
(all x all y all z (T(x) & T(y) & T(z) -> (SUM(z,x,y) <-> (all w (T(w) -> (O(w,z) <-> O(w,x) | O(w,y)))))))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/

dolce_taxonomy.p9
(all x (ED(x) | PD(x) | Q(x) | AB(x) -> PT(x))).
(all x (PED(x) | NPED(x) | AS(x) -> ED(x))).
(all x (EV(x) | STV(x) -> PD(x))).
(all x (TQ(x) | PQ(x) | AQ(x) -> Q(x))).
(all x (R(x) -> AB(x))).
(all x (M(x) | F(x) | POB(x) -> PED(x))).
(all x (NPOB(x) -> NPED(x))).
(all x (ACH(x) | ACC(x) -> EV(x))).
(all x (ST(x) | PRO(x) -> STV(x))).
(all x (TL(x) -> TQ(x))).
(all x (SL(x) -> PQ(x))).
(all x (TR(x) | PR(x) | AR(x) -> R(x))).
(all x (APO(x) | NAPO(x) -> POB(x))).
(all x (MOB(x) | SOB(x) -> NPOB(x))).
(all x (T(x) -> TR(x))).
(all x (S(x) -> PR(x))).
(all x (ASO(x) | NASO(x) -> SOB(x))).
(all x (SAG(x) | SC(x) -> ASO(x))).
(all x (PT(x) <-> ED(x) | PD(x) | Q(x) | AB(x))).
(all x (ED(x) -> -PD(x) & -Q(x) & -AB(x))).
(all x (PD(x) -> -Q(x) & -AB(x))).
(all x (Q(x) -> -AB(x))).
(all x (ED(x) <-> PED(x) | NPED(x) | AS(x))).
(all x (PED(x) -> -NPED(x) & -AS(x))).
(all x (NPED(x) -> -AS(x))).
(all x (PD(x) <-> EV(x) | STV(x))).
(all x (EV(x) -> -STV(x))).
(all x (Q(x) <-> TQ(x) | PQ(x) | AQ(x))).
(all x (TQ(x) -> -PQ(x) & -AQ(x))).
(all x (PQ(x) -> -AQ(x))).
(all x (PED(x) <-> M(x) | F(x) | POB(x))).
(all x (M(x) -> -F(x) & -POB(x))).
(all x (F(x) -> -POB(x))).
(all x (EV(x) <-> ACH(x) | ACC(x))).
(all x (ACH(x) -> -ACC(x))).
(all x (STV(x) <-> ST(x) | PRO(x))).
(all x (ST(x) -> -PRO(x))).
(all x (R(x) <-> TR(x) | PR(x) | AR(x))).
(all x (TR(x) -> -PR(x) & -AR(x))).
(all x (PR(x) -> -AR(x))).
(all x (POB(x) <-> APO(x) | NAPO(x))).
(all x (APO(x) -> -NAPO(x))).
(all x (NPOB(x) <-> MOB(x) | SOB(x))).
(all x (MOB(x) -> -SOB(x))).
(all x (SOB(x) <-> ASO(x) | NASO(x))).
(all x (ASO(x) -> -NASO(x))).
(all x (ASO(x) <-> SAG(x) | SC(x))).
(all x (SAG(x) -> -SC(x))).

%*****relevance mappings to requirements
%dolce_2_tr
%katsumi-thesis-searchcase-tr-to-dolce_participation-translation-defs
% 'PC equivalent to tr_participates_in'
(all x all a all t (PC(x,a,t) <-> tr_participates_in(x,a,t))).

%'PD equivalent to tr_event'
(all a (PD(a) <-> tr_event(a))).

%'no dolce_participation equivalent for tr_has_role'

%'PRE equivalent to tr_exists_at'
(all a all t (PRE(a,t) <-> tr_exists_at(a,t))).

%'no dolce_participation equivalent for tr_part_of'

%'no dolce_participation equivalent for tr_object but we are given the additional info that an object is an endurant'
%(all a (tr_object(a) -> ED(a))).

%'no dolce_participation equivalent for tr_active_in'
%'no dolce_participation equivalent for tr_passive_in'

%'no dolce_participation equivalent for tr_consumed_in'

%*********root theory
%Root Theory (collection of candidates' signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t) )
)).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))
).
(all a all t (tr_exists_at(x,t) -> (tr_exists_at(x,t))
).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))
).
(all x (tr_object(x) -> tr_object(x))
).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))
).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))
).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))
).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))
).
(all x (g_Object(x) -> g_Object(x))
).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))
).
(all x all y (g_pre(x,y) -> g_pre(x,y))
).
(all y (g_Time(y) -> g_Time(y))
).

%ps1_participation.p9 signature
%psl_prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))
).
(all x (psl_object(x) -> psl_object(x))
).
(all x (psl_timepoint(x) -> psl_timepoint(x))
).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))
).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))
).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x ,occ,t))
).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))
).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))
).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))
).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))
).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))
).
(all x psl_endof(x) = psl_endof(x)).
Appendix B. Comparing Participation Ontologies: Proof Files

(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ5 'We need to talk about different types of participation: active, passive, consumption.'

-((all x all a all t
    (tr_participates_in(x,a,t)
    ->
        (tr_active_in(x,a,t) | tr_passive_in(x,a,t) |
        tr_consumed_in(x,a,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 3

c1 = 0.
c2 = 1.
c3 = 2.

psl_beginof(0) = 0.
psl_beginof(1) = 0.
psl_beginof(2) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.
psl_endof(2) = 0.

f2(0) = 1.
f2(1) = 0.
f2(2) = 0.

f3(0) = 2.
f3(1) = 0.
f3(2) = 0.

f5(0) = 2.
f5(1) = 2.
f5(2) = 0.

f10(0) = 0.
f10(1) = 0.
f10(2) = 0.

f1(0,0) = 0.
f1(0,1) = 0.
f1(0,2) = 0.
f1(1,0) = 0.
f1(1,1) = 0.
f1(1,2) = 0.
f1(2,0) = 0.
f1(2,1) = 0.
f1(2,2) = 0.

f6(0,0) = 0.
f6(0,1) = 0.
f6(0,2) = 0.
f6(1,0) = 0.
f6(1,1) = 0.
f6(1,2) = 0.
f6(2,0) = 0.
f6(2,1) = 0.
f6(2,2) = 0.
\[ f_7(0,0) = 0. \]
\[ f_7(0,1) = 0. \]
\[ f_7(0,2) = 0. \]
\[ f_7(1,0) = 0. \]
\[ f_7(1,1) = 0. \]
\[ f_7(1,2) = 0. \]
\[ f_7(2,0) = 0. \]
\[ f_7(2,1) = 0. \]
\[ f_7(2,2) = 0. \]

\[ f_8(0,0) = 0. \]
\[ f_8(0,1) = 0. \]
\[ f_8(0,2) = 0. \]
\[ f_8(1,0) = 0. \]
\[ f_8(1,1) = 0. \]
\[ f_8(1,2) = 0. \]
\[ f_8(2,0) = 0. \]
\[ f_8(2,1) = 0. \]
\[ f_8(2,2) = 2. \]

\[ f_9(0,0) = 0. \]
\[ f_9(0,1) = 0. \]
\[ f_9(0,2) = 0. \]
\[ f_9(1,0) = 0. \]
\[ f_9(1,1) = 0. \]
\[ f_9(1,2) = 0. \]
\[ f_9(2,0) = 0. \]
\[ f_9(2,1) = 0. \]
\[ f_9(2,2) = 2. \]

\[ f_{11}(0,0) = 0. \]
\[ f_{11}(0,1) = 0. \]
\[ f_{11}(0,2) = 0. \]
\[ f_{11}(1,0) = 0. \]
\[ f_{11}(1,1) = 0. \]
\[ f_{11}(1,2) = 0. \]
\[ f_{11}(2,0) = 0. \]
\[ f_{11}(2,1) = 0. \]
\[ f_{11}(2,2) = 2. \]

\[ f_{12}(0,0) = 0. \]
\[ f_{12}(0,1) = 0. \]
\[ f_{12}(0,2) = 0. \]
\[ f_{12}(1,0) = 0. \]
\[ f_{12}(1,1) = 0. \]
\[ f_{12}(1,2) = 0. \]
\[ f_{12}(2,0) = 0. \]
\[ f_{12}(2,1) = 0. \]
\[ f_{12}(2,2) = 2. \]

\[ f_{4}(0,0,0) = 0. \]
\[ f_{4}(0,0,1) = 0. \]
\[ f_{4}(0,0,2) = 0. \]
\[ f_{4}(0,1,0) = 0. \]
\[ f_{4}(0,1,1) = 0. \]
\[ f_{4}(0,1,2) = 0. \]
\[ f_{4}(0,2,0) = 0. \]
\[ f_{4}(0,2,1) = 0. \]
\[ f_{4}(0,2,2) = 0. \]
\[ f_{4}(1,0,0) = 0. \]
\[ f_{4}(1,0,1) = 0. \]
\[ f_{4}(1,0,2) = 0. \]
\[ f_{4}(1,1,0) = 0. \]
\[ f_{4}(1,1,1) = 0. \]
\[ f_{4}(1,1,2) = 0. \]
\[ f_{4}(1,2,0) = 0. \]
\[ f_{4}(1,2,1) = 0. \]
\[ f_{4}(1,2,2) = 0. \]
\[ f_{4}(2,0,0) = 0. \]
\[ f_{4}(2,0,1) = 0. \]
\[ f_{4}(2,0,2) = 0. \]
\[ f_{4}(2,1,0) = 0. \]
\[ f_{4}(2,1,1) = 0. \]
\[ f_{4}(2,1,2) = 0. \]
\[ f_{4}(2,2,0) = 0. \]
\[ f_{4}(2,2,1) = 0. \]
\[ f_{4}(2,2,2) = 0. \]

\[ f_{13}(0,0,0) = 0. \]
\[ f_{13}(0,0,1) = 0. \]
\[ f_{13}(0,0,2) = 0. \]
\[ f_{13}(0,1,0) = 0. \]
\[ f_{13}(0,1,1) = 0. \]
\[ f_{13}(0,1,2) = 0. \]
\[ f_{13}(0,2,0) = 0. \]
\[ f_{13}(0,2,1) = 0. \]
\begin{verbatim}
  f13(0,2,2) = 0.
  f13(1,0,0) = 0.
  f13(1,0,1) = 0.
  f13(1,0,2) = 0.
  f13(1,1,0) = 0.
  f13(1,1,1) = 0.
  f13(1,1,2) = 0.
  f13(1,2,0) = 0.
  f13(1,2,1) = 0.
  f13(1,2,2) = 0.
  f13(2,0,0) = 0.
  f13(2,0,1) = 0.
  f13(2,0,2) = 0.
  f13(2,1,0) = 0.
  f13(2,1,1) = 0.
  f13(2,1,2) = 0.
  f13(2,2,0) = 0.
  f13(2,2,1) = 0.
  f13(2,2,2) = 0.

- AB(0).
- AB(1).
  AB(2).

- ACC(0).
- ACC(1).
- ACC(2).

- ACH(0).
- ACH(1).
- ACH(2).

- APO(0).
- APO(1).
- APO(2).

- AQ(0).
- AQ(1).
- AQ(2).

- AR(0).
- AR(1).
- AR(2).
\end{verbatim}
- AS(0).
- AS(1).
- AS(2).

- ASO(0).
- ASO(1).
- ASO(2).

- AtP(0).
- AtP(1).
- AtP(2).

- ED(0).
- ED(1).
- ED(2).

- EV(0).
- EV(1).
- EV(2).

- F(0).
- F(1).
- F(2).

- M(0).
- M(1).
- M(2).

- MOB(0).
- MOB(1).
- MOB(2).

- NAPO(0).
- NAPO(1).
- NAPO(2).

- NASO(0).
- NASO(1).
- NASO(2).

- NPED(0).
- NPED(1).
- NPED(2).
- NPOB(0).
- NPOB(1).
- NPOB(2).
- PD(0).
  PD(1).
- PD(2).
- PED(0).
- PED(1).
- PED(2).
- POB(0).
- POB(1).
- POB(2).
- PQ(0).
- PQ(1).
- PQ(2).
- PR(0).
- PR(1).
- PR(2).
- PRO(0).
- PRO(1).
- PRO(2).
  PT(0).
  PT(1).
- PT(2).
- Q(0).
- Q(1).
- Q(2).
- R(0).
- R(1).
- R(2).
- S(0).
- S(1).
- S(2).

- SAG(0).
- SAG(1).
- SAG(2).

- SC(0).
- SC(1).
- SC(2).

- SL(0).
- SL(1).
- SL(2).

- SOB(0).
- SOB(1).
- SOB(2).

- ST(0).
  ST(1).
- ST(2).

- STV(0).
  STV(1).
- STV(2).

- T(0).
- T(1).
  T(2).

- TL(0).
- TL(1).
- TL(2).

- TQ(0).
- TQ(1).
- TQ(2).

- TR(0).
- TR(1).
- TR(2).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES 450

- tr_event(0).
  tr_event(1).
- tr_event(2).

- DJ(0,0).
- DJ(0,1).
- DJ(0,2).
- DJ(1,0).
- DJ(1,1).
- DJ(1,2).
- DJ(2,0).
- DJ(2,1).
- DJ(2,2).

- O(0,0).
- O(0,1).
- O(0,2).
- O(1,0).
- O(1,1).
- O(1,2).
- O(2,0).
- O(2,1).
- O(2,2).

- P(0,0).
- P(0,1).
- P(0,2).
- P(1,0).
- P(1,1).
- P(1,2).
- P(2,0).
- P(2,1).
- P(2,2).

- PP(0,0).
- PP(0,1).
- PP(0,2).
- PP(1,0).
- PP(1,1).
- PP(1,2).
- PP(2,0).
- PP(2,1).
- PP(2,2).
- PRE(0,0).
- PRE(0,1).
- PRE(0,2).
- PRE(1,0).
- PRE(1,1).
- PRE(1,2).
- PRE(2,0).
- PRE(2,1).
- PRE(2,2).

- U(0,0).
- U(0,1).
- U(0,2).
- U(1,0).
- U(1,1).
- U(1,2).
- U(2,0).
- U(2,1).
- U(2,2).

- tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(0,2).
- tr_exists_at(1,0).
- tr_exists_at(1,1).
- tr_exists_at(1,2).
- tr_exists_at(2,0).
- tr_exists_at(2,1).
- tr_exists_at(2,2).

- PC(0,0,0).
- PC(0,0,1).
- PC(0,0,2).
- PC(0,1,0).
- PC(0,1,1).
- PC(0,1,2).
- PC(0,2,0).
- PC(0,2,1).
- PC(0,2,2).
- PC(1,0,0).
- PC(1,0,1).
- PC(1,0,2).


- PC(1,1,0).
- PC(1,1,1).
- PC(1,1,2).
- PC(1,2,0).
- PC(1,2,1).
- PC(1,2,2).
- PC(2,0,0).
- PC(2,0,1).
- PC(2,0,2).
- PC(2,1,0).
- PC(2,1,1).
- PC(2,1,2).
- PC(2,2,0).
- PC(2,2,1).
- PC(2,2,2).

- SUM(0,0,0).
- SUM(0,0,1).
- SUM(0,0,2).
- SUM(0,1,0).
- SUM(0,1,1).
- SUM(0,1,2).
- SUM(0,2,0).
- SUM(0,2,1).
- SUM(0,2,2).
- SUM(1,0,0).
- SUM(1,0,1).
- SUM(1,0,2).
- SUM(1,1,0).
- SUM(1,1,1).
- SUM(1,1,2).
- SUM(1,2,0).
- SUM(1,2,1).
- SUM(1,2,2).
- SUM(2,0,0).
- SUM(2,0,1).
- SUM(2,0,2).
- SUM(2,1,0).
- SUM(2,1,1).
- SUM(2,1,2).
- SUM(2,2,0).
- SUM(2,2,1).
- SUM(2,2,2).
- tr_active_in(0,0,0).
- tr_active_in(0,0,1).
- tr_active_in(0,0,2).
- tr_active_in(0,1,0).
- tr_active_in(0,1,1).
- tr_active_in(0,1,2).
- tr_active_in(0,2,0).
- tr_active_in(0,2,1).
- tr_active_in(0,2,2).
- tr_active_in(1,0,0).
- tr_active_in(1,0,1).
- tr_active_in(1,0,2).
- tr_active_in(1,1,0).
- tr_active_in(1,1,1).
- tr_active_in(1,1,2).
- tr_active_in(1,2,0).
- tr_active_in(1,2,1).
- tr_active_in(1,2,2).
- tr_active_in(2,0,0).
- tr_active_in(2,0,1).
- tr_active_in(2,0,2).
- tr_active_in(2,1,0).
- tr_active_in(2,1,1).
- tr_active_in(2,1,2).
- tr_active_in(2,2,0).
- tr_active_in(2,2,1).
- tr_active_in(2,2,2).

- tr_consumed_in(0,0,0).
- tr_consumed_in(0,0,1).
- tr_consumed_in(0,0,2).
- tr_consumed_in(0,1,0).
- tr_consumed_in(0,1,1).
- tr_consumed_in(0,1,2).
- tr_consumed_in(0,2,0).
- tr_consumed_in(0,2,1).
- tr_consumed_in(0,2,2).
- tr_consumed_in(1,0,0).
- tr_consumed_in(1,0,1).
- tr_consumed_in(1,0,2).
- tr_consumed_in(1,1,0).
- tr_consumed_in(1,1,1).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[- \text{tr\_consumed\_in}(1,1,2).\]
\[- \text{tr\_consumed\_in}(1,2,0).\]
\[- \text{tr\_consumed\_in}(1,2,1).\]
\[- \text{tr\_consumed\_in}(1,2,2).\]
\[- \text{tr\_consumed\_in}(2,0,0).\]
\[- \text{tr\_consumed\_in}(2,0,1).\]
\[- \text{tr\_consumed\_in}(2,0,2).\]
\[- \text{tr\_consumed\_in}(2,1,0).\]
\[- \text{tr\_consumed\_in}(2,1,1).\]
\[- \text{tr\_consumed\_in}(2,1,2).\]
\[- \text{tr\_consumed\_in}(2,2,0).\]
\[- \text{tr\_consumed\_in}(2,2,1).\]
\[- \text{tr\_consumed\_in}(2,2,2).\]

\[- \text{tr\_participates\_in}(0,0,0).\]
\[- \text{tr\_participates\_in}(0,0,1).\]
\[- \text{tr\_participates\_in}(0,0,2).\]
\[- \text{tr\_participates\_in}(0,1,0).\]
\[- \text{tr\_participates\_in}(0,1,1).\]
\[- \text{tr\_participates\_in}(0,1,2).\]
\[- \text{tr\_participates\_in}(0,2,0).\]
\[- \text{tr\_participates\_in}(0,2,1).\]
\[- \text{tr\_participates\_in}(0,2,2).\]
\[- \text{tr\_participates\_in}(1,0,0).\]
\[- \text{tr\_participates\_in}(1,0,1).\]
\[- \text{tr\_participates\_in}(1,0,2).\]
\[- \text{tr\_participates\_in}(1,1,0).\]
\[- \text{tr\_participates\_in}(1,1,1).\]
\[- \text{tr\_participates\_in}(1,1,2).\]
\[- \text{tr\_participates\_in}(1,2,0).\]
\[- \text{tr\_participates\_in}(1,2,1).\]
\[- \text{tr\_participates\_in}(1,2,2).\]
\[- \text{tr\_participates\_in}(2,0,0).\]
\[- \text{tr\_participates\_in}(2,0,1).\]
\[- \text{tr\_participates\_in}(2,0,2).\]
\[- \text{tr\_participates\_in}(2,1,0).\]
\[- \text{tr\_participates\_in}(2,1,1).\]
\[- \text{tr\_participates\_in}(2,1,2).\]
\[- \text{tr\_participates\_in}(2,2,0).\]
\[- \text{tr\_participates\_in}(2,2,1).\]
\[- \text{tr\_participates\_in}(2,2,2).\]

\[- \text{tr\_passive\_in}(0,0,0).\]
B.4.3 $H_{gangemi\_participation}$

$H_{gangemi\_participation} \neq CQ2$ Proof attempt for CQ2:

```prolog
%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/gangemi.
   p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (g_Object(x) | g_Event(x)) & g_Time(y))).

%relevance mappings to requirements (all):
```
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t)))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory:%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
Appendix B. Comparing Participation Ontologies: Proof Files

(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

%dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

%dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

%dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ2 'An object must exist in order for it to participate in some event.'
-((all x all a all t
   (tr_participates_in(x,a,t)
     & tr_event(a)
     ->
     tr_exists_at(x,t)))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.

c1 = 0.
c2 = 0.
c3 = 1.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.

f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0,0) = 0.
f5(0,1) = 0.
f5(1,0) = 0.
f5(1,1) = 0.

g_Event(0).
g_Object(0).
g_Time(0).

tr_event(0).
tr_object(0).

- g_Event(1).
- g_Object(1).
- g_Time(1).
- tr_event(1).
- tr_object(1).
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[
g_{\text{hasParticipant}}(0,0).
- g_{\text{hasParticipant}}(0,1).
- g_{\text{hasParticipant}}(1,0).
- g_{\text{hasParticipant}}(1,1).
\]

\[
g_{\text{pre}}(0,0).
\quad g_{\text{pre}}(0,1).
- g_{\text{pre}}(1,0).
- g_{\text{pre}}(1,1).
\]

\[
\text{tr}_{\text{exists_at}}(0,0).
- \text{tr}_{\text{exists_at}}(0,1).
- \text{tr}_{\text{exists_at}}(1,0).
- \text{tr}_{\text{exists_at}}(1,1).
\]

\[
\text{tr}_{\text{participates_in}}(0,0,0).
\quad \text{tr}_{\text{participates_in}}(0,0,1).
- \text{tr}_{\text{participates_in}}(0,1,0).
- \text{tr}_{\text{participates_in}}(0,1,1).
- \text{tr}_{\text{participates_in}}(1,0,0).
- \text{tr}_{\text{participates_in}}(1,0,1).
- \text{tr}_{\text{participates_in}}(1,1,0).
- \text{tr}_{\text{participates_in}}(1,1,1).
\]

\[
H_{\text{gangemi.participation}} \not\equiv CQ3 \text{ Proof attempt for CQ3:}
\]

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/gangemi.p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_{\text{Event}}(x) -> (exists y (g_{\text{Object}}(x) & g_{\text{hasParticipant}}(x,y))))).
(all x (g_{\text{Object}}(x) -> (exists y (g_{\text{Event}}(x) & g_{\text{hasParticipant}}(y,x))))).
(all x (g_{\text{Event}}(x) -> (exists y g_{\text{pre}}(x,y))))).
(all x (g_{\text{Object}}(x) -> (exists y g_{\text{pre}}(x,y))))).
(all x all y (g_{\text{pre}}(x,y) -> (g_{\text{Object}}(x) | g_{\text{Event}}(x)) & g_{\text{Time}}(y))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_{\text{Event}}(x) <-> \text{tr}_{\text{event}}(x))).
(all x (g_{\text{Object}}(x) <-> \text{tr}_{\text{object}}(x))).

(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t))))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory:
% Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
% psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))).
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).
%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
   dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ3 ’If something participates in an activity, then this participation translates to any other event that the activity may be a part of.’

-((all x all a_1 all a_2 all t
(tr_participates_in(x,a_1,t)
 & tr_part_of(a_1,a_2)
 ->
 tr_participates_in(x,a_2,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.

c1 = 0.

c2 = 0.

c3 = 1.
APPENDIX B. COMPARING PARTICIPATION ONTOLOGIES: PROOF FILES

\[ c_4 = 0. \]

\[
\text{psl\_beginof}(0) = 0. \\
\text{psl\_beginof}(1) = 0. \\
\text{psl\_endof}(0) = 0. \\
\text{psl\_endof}(1) = 0. \\
\]

\[
\text{f}_1(0) = 0. \\
\text{f}_1(1) = 0. \\
\text{f}_2(0) = 0. \\
\text{f}_2(1) = 0. \\
\text{f}_3(0) = 0. \\
\text{f}_3(1) = 0. \\
\text{f}_4(0) = 0. \\
\text{f}_4(1) = 0. \\
\text{f}_5(0,0) = 0. \\
\text{f}_5(0,1) = 0. \\
\text{f}_5(1,0) = 0. \\
\text{f}_5(1,1) = 0. \\
\]

- \text{g\_Event}(0).
- \text{g\_Event}(1).
- \text{g\_Object}(0).
- \text{g\_Object}(1).
- \text{g\_Time}(0).
- \text{g\_Time}(1).
- \text{tr\_event}(0).
- \text{tr\_event}(1).
- \text{tr\_object}(0).
- \text{tr\_object}(1).
- \text{g\_hasParticipant}(0,0).
- \text{g\_hasParticipant}(0,1).
- \text{g\_hasParticipant}(1,0).
- \texttt{g\_hasParticipant(1,1)}.
- \texttt{g\_pre(0,0)}.
- \texttt{g\_pre(0,1)}.
- \texttt{g\_pre(1,0)}.
- \texttt{g\_pre(1,1)}.

- \texttt{tr\_exists\_at(0,0)}.
- \texttt{tr\_exists\_at(0,1)}.
- \texttt{tr\_exists\_at(1,0)}.
- \texttt{tr\_exists\_at(1,1)}.

- \texttt{tr\_part\_of(0,0)}.
- \texttt{tr\_part\_of(0,1)}.
- \texttt{tr\_part\_of(1,0)}.
- \texttt{tr\_part\_of(1,1)}.

- \texttt{tr\_participates\_in(0,0,0)}.
- \texttt{tr\_participates\_in(0,0,1)}.
- \texttt{tr\_participates\_in(0,1,0)}.
- \texttt{tr\_participates\_in(0,1,1)}.
- \texttt{tr\_participates\_in(1,0,0)}.
- \texttt{tr\_participates\_in(1,0,1)}.
- \texttt{tr\_participates\_in(1,1,0)}.
- \texttt{tr\_participates\_in(1,1,1)}.

\section*{Proof attempt for CQ4:}

\begin{verbatim}
%common hierarchy  % Reading from file /stl/chui/macleod-master/qs/test/conversions/gangemi.
p9  % g_ prefix added to all terms to distinguish from other candidates

(all x (g\_Event(x) -> (exists y (g\_Object(x) & g\_hasParticipant(x,y))))).
(all x (g\_Object(x) -> (exists y (g\_Event(x) & g\_hasParticipant(y,x))))).
(all x (g\_Event(x) -> (exists y g\_pre(x,y))))).
(all x (g\_Object(x) -> (exists y g\_pre(x,y))))).
(all x all y (g\_pre(x,y) -> (g\_Object(x) \| g\_Event(x) & g\_Time(y))))).

%relevance mappings to requirements (all):
%gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates
\end{verbatim}
(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t)))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory:
%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occurring_at(o,t) -> psl_is_occurring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3)))


(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
  dolce_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ARCH(x) -> ARCH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).
(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
% CQ4 'Objects can not participate in more than one different activity.'
-((all x all a_1 all a_2
  (tr_object(x)
    & tr_participates_in(x,a_1,t)
    & tr_participates_in(x,a_2,t)
    ->
      ((a_1 = a_2) | tr_part_of(a_1,a_2)))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

t = 0.

c1 = 0.

c2 = 0.
c3 = 1.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.

f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0,0) = 0.
f5(0,1) = 0.
f5(1,0) = 0.
f5(1,1) = 0.

g_Event(0).
- g_Event(1).

g_Object(0).
- g_Object(1).

g_Time(0).
g_Time(1).

tr_event(0).
- tr_event(1).

tr_object(0).
- tr_object(1).

g_hasParticipant(0,0).
- g_hasParticipant(0,1).
- g_hasParticipant(1,0).
- g_hasParticipant(1,1).
   g_pre(0,0).
   g_pre(0,1).
- g_pre(1,0).
- g_pre(1,1).

   tr_exists_at(0,0).
- tr_exists_at(0,1).
- tr_exists_at(1,0).
- tr_exists_at(1,1).

- tr_part_of(0,0).
- tr_part_of(0,1).
- tr_part_of(1,0).
- tr_part_of(1,1).

   tr_participates_in(0,0,0).
- tr_participates_in(0,0,1).
   tr_participates_in(0,1,0).
- tr_participates_in(0,1,1).
- tr_participates_in(1,0,0).
- tr_participates_in(1,0,1).
- tr_participates_in(1,1,0).
- tr_participates_in(1,1,1).

Proof attempt for CQ5:

%common hierarchy
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/gangemi.
p9
% g_ prefix added to all terms to distinguish from other candidates

(all x (g_Event(x) -> (exists y (g_Object(x) & g_hasParticipant(x,y))))).
(all x (g_Object(x) -> (exists y (g_Event(x) & g_hasParticipant(y,x))))).
(all x (g_Event(x) -> (exists y g_pre(x,y))))).
(all x (g_Object(x) -> (exists y g_pre(x,y))))).
(all x all y (g_pre(x,y) -> (g_Object(x) | g_Event(x)) & g_Time(y)))).

%relevance mappings to requirements (all):
gangemi2Tr
% translation from gangemi.p9 to requirements signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) <-> tr_event(x))).
(all x (g_Object(x) <-> tr_object(x))).
(all x all y (g_hasParticipant(x,y) <-> (exists t (tr_participates_in(y,x,t)))).
(all x all y (g_pre(x,y) <-> tr_exists_at(x,t))).

%root theory:%Root Theory (collection of candidates’ signature term tautologies)
% katsumi-thesis-searchcase-semantic-reqts-v1.clif signature
(all x all a all t (tr_participates_in(x,a,t) -> (tr_participates_in(x,a,t)))).
(all a (tr_event(a) -> tr_event(a))).
(all x all z all t (tr_has_role(x,z,t) -> (tr_has_role(x,z,t)))).
(all x all t (tr_exists_at(x,t) -> (tr_exists_at(x,t)))).
(all a_1 all a_2 (tr_part_of(a_1,a_2) -> tr_part_of(a_1,a_2))).
(all x (tr_object(x) -> tr_object(x))).
(all x all a all t (tr_active_in(x,a,t) -> tr_active_in(x,a,t))).
(all x all a all t (tr_passive_in(x,a,t) -> tr_passive_in(x,a,t))).
(all x all a all t (tr_consumed_in(x,a,t) -> tr_consumed_in(x,a,t))).

%gangemi.p9 signature
% g_ prefix added to all terms to distinguish from other candidates
(all x (g_Event(x) -> g_Event(x))).
(all x (g_Object(x) -> g_Object(x))).
(all x all y (g_hasParticipant(x,y) -> g_hasParticipant(x,y))).
(all x all y (g_pre(x,y) -> g_pre(x,y))).
(all y (g_Time(y) -> g_Time(y))).

%psl_participation.p9 signature
%psl_ prefix added to all terms to distinguish from other candidates
(all x (psl_activity_occurrence(x) -> psl_activity_occurrence(x))).
(all x (psl_object(x) -> psl_object(x))).
(all x (psl_timepoint(x) -> psl_timepoint(x))).
(all o all t (psl_is_occuring_at(o,t) -> psl_is_occuring_at(o,t))).
(all x all t (psl_exists_at(x,t) -> psl_exists_at(x,t))).
(all x all occ all t (psl_participates_in(x,occ,t) -> psl_participates_in(x,occ,t))).
(all t1 all t2 all t3 (psl_between(t1,t2,t3) -> psl_between(t1,t2,t3))).
(all t1 all t2 (psl_before(t1,t2) -> psl_before(t1,t2))).
(all t1 all t2 (psl_beforeEq(t1,t2) -> psl_beforeEq(t1,t2))).
(all t1 (psl_timepoint(t1) -> psl_timepoint(t1))).
(all t1 all t2 all t3 (psl_betweenEq(t1,t2,t3) -> psl_betweenEq(t1,t2,t3))) .
(all x psl_endof(x) = psl_endof(x)).
(all x psl_beginof(x) = psl_beginof(x)).

%dolce_participation.p9 signature
% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
daôle_participation.p9
(all x all y all t (PC(x,y,t) -> PC(x,y,t))).
(all x (ED(x) -> ED(x))).
(all y (PD(y) -> PD(y))).
(all t (T(t) -> T(t))).
(all x all t (PRE(x,t) -> PRE(x,t))).
(all t1 all t (P(t1,t) -> P(t1,t))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
daôle_present.p9
(all x (Q(x) -> Q(x))).
(all t all t1 all t2 (SUM(t,t1,t2) -> SUM(t,t1,t2))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
daôle_time_mereology.p9
(all z all y (O(z,y) -> O(z,y))).
(all z all y (DJ(z,y) -> DJ(z,y))).
(all x all y (PP(x,y) -> PP(x,y))).
(all x all y (U(x,y) -> U(x,y))).
(all x (AtP(x) -> AtP(x))).
(all x all y (U(x,y) -> U(x,y))).

% Reading from file /stl/cchui/macleod-master/qs/test/conversions/
daôle_taxonomy.p9
(all x (AB(x) -> AB(x))).
(all x (PED(x) -> PED(x))).
(all x (NPED(x) -> NPED(x))).
(all x (AS(x) -> AS(x))).
(all x (EV(x) -> EV(x))).
(all x (STV(x) -> STV(x))).
(all x (TQ(x) -> TQ(x))).
(all x (PQ(x) -> PQ(x))).
(all x (AQ(x) -> AQ(x))).
(all x (R(x) -> R(x))).
(all x (M(x) -> M(x))).
(all x (F(x) -> F(x))).
(all x (POB(x) -> POB(x))).
(all x (NPOB(x) -> NPOB(x))).
(all x (ACH(x) -> ACH(x))).
(all x (ACC(x) -> ACC(x))).
(all x (ST(x) -> ST(x))).
(all x (PRO(x) -> PRO(x))).
(all x (TL(x) -> TL(x))).
(all x (SL(x) -> SL(x))).
(all x (TR(x) -> TR(x))).
(all x (PR(x) -> PR(x))).
(all x (AR(x) -> AR(x))).
(all x (APO(x) -> APO(x))).
(all x (NAPO(x) -> NAPO(x))).
(all x (MOB(x) -> MOB(x))).
(all x (SOB(x) -> SOB(x))).

(all x (S(x) -> S(x))).
(all x (ASO(x) -> ASO(x))).
(all x (NASO(x) -> NASO(x))).
(all x (SAG(x) -> SAG(x))).
(all x (SC(x) -> SC(x))).
(all x (PT(x) -> PT(x))).
(all x (PD(x) -> PD(x))).

%goal
%CQ5 'We need to talk about different types of participation: active, passive, consumption.'
-
(all x all a all t
  (tr_participates_in(x,a,t)
  ->
   (tr_active_in(x,a,t) | tr_passive_in(x,a,t) | tr_consumed_in(x,a,t))))).

Counterexample found:

% number = 1
% seconds = 0

% Interpretation of size 2

  t = 0.
  c1 = 0.
  c2 = 0.
c3 = 0.

psl_beginof(0) = 0.
psl_beginof(1) = 0.

psl_endof(0) = 0.
psl_endof(1) = 0.

f1(0) = 0.
f1(1) = 0.

f2(0) = 0.
f2(1) = 0.

f3(0) = 0.
f3(1) = 0.

f4(0) = 0.
f4(1) = 0.

f5(0, 0) = 0.
f5(0, 1) = 0.
f5(1, 0) = 0.
f5(1, 1) = 0.

- g_Event(0).
- g_Event(1).

- g_Object(0).
- g_Object(1).

- g_Time(0).
- g_Time(1).

- tr_event(0).
- tr_event(1).

- tr_object(0).
- tr_object(1).

  g_hasParticipant(0, 0).
- g_hasParticipant(0, 1).
- g_hasParticipant(1, 0).
- g\_hasParticipant(1,1).

- g\_pre(0,0).
- g\_pre(0,1).
- g\_pre(1,0).
- g\_pre(1,1).

- tr\_exists\_at(0,0).
- tr\_exists\_at(0,1).
- tr\_exists\_at(1,0).
- tr\_exists\_at(1,1).

- tr\_active\_in(0,0,0).
- tr\_active\_in(0,0,1).
- tr\_active\_in(0,1,0).
- tr\_active\_in(0,1,1).
- tr\_active\_in(1,0,0).
- tr\_active\_in(1,0,1).
- tr\_active\_in(1,1,0).
- tr\_active\_in(1,1,1).

- tr\_consumed\_in(0,0,0).
- tr\_consumed\_in(0,0,1).
- tr\_consumed\_in(0,1,0).
- tr\_consumed\_in(0,1,1).
- tr\_consumed\_in(1,0,0).
- tr\_consumed\_in(1,0,1).
- tr\_consumed\_in(1,1,0).
- tr\_consumed\_in(1,1,1).

tr\_participates\_in(0,0,0).
- tr\_participates\_in(0,0,1).
- tr\_participates\_in(0,1,0).
- tr\_participates\_in(0,1,1).
- tr\_participates\_in(1,0,0).
- tr\_participates\_in(1,0,1).
- tr\_participates\_in(1,1,0).
- tr\_participates\_in(1,1,1).

- tr\_passive\_in(0,0,0).
- tr\_passive\_in(0,0,1).
- tr\_passive\_in(0,1,0).
- tr\_passive\_in(0,1,1).
- tr\_passive\_in(1,0,0).
- tr\_passive\_in(1,0,1).
- tr\_passive\_in(1,1,0).
- tr\_passive\_in(1,1,1).
- tr_passive_in(1,0,0).
- tr_passive_in(1,0,1).
- tr_passive_in(1,1,0).
- tr_passive_in(1,1,1).
Appendix C

Motivation for Reuse with Translations

C.1 Signature Translation: Identifying Reuse of Ontology Patterns

Here we show how a single theory can be reused in multiple domains with only a translation of its signature, essentially performing the role of an ontology design pattern (ODP). This material first appeared in [39]. Within COLORE, design patterns are formalized as core ontologies within the repository. Patterns are reused via the metatheoretic relationships of relative interpretation and definable equivalence, in other words this work looks at the reuse of design patterns via signature translation. In this sense, the ontology design patterns within COLORE are semantic (model-theoretic) rather than syntactic. On the other hand, the approach described here can also be used to generate axioms for new ontologies, in which case we can consider core ontologies to serve as syntactic templates for axioms.

After an informal discussion of ontology design patterns in the context of COLORE, an overview of the relationships between ontologies within COLORE is given. The notions of relative interpretation, definable equivalence, and reduction played a key role in formalizing the reuse of ontologies. In particular, these notions provided techniques for evaluating ontology design patterns and proving that a pattern is correctly and completely exemplified by a set of ontologies. This approach was illustrated using sets of ontologies within COLORE.

C.1.1 COLORE and Ontology Design Patterns

In general, Ontology Design Patterns (OPs) are meant to serve as reusable solutions for various aspects of ontology design [24], and the structure of the ontologies in COLORE and the relationships defined between them can provide similar support. COLORE provides a means of sharing content ontology design patterns (CPs) while providing solutions that address specific instances of some of the modelling problems that other OPs are designed to solve.

Of the six families of OPs recognized in [24], the Structural, Correspondence, and Content families of OPs have strong parallels in COLORE:

Structural OPs include what are referred to as Logical and Architectural OPs. Architectural OPs represent possible structures for an ontology being designed. These structures are meant to assist with design choices when computational complexity is a concern, and also to serve as reference material to guide designers in creating their own structures. In particular, external Architectural OPs provide patterns for ontology modularization, (“meta-level constructs”). Examples of these external Architectural OPs can be found in
Appendix C. Motivation for Reuse with Translations

COLORE as each ontology is stored in modules[^36] that are connected to form the ontology using the imports relation.

**Correspondence OPs** include what are referred to as Reengineering and Mapping OPs. Mapping OPs provide a means to describe the relationship(s) that exist between elements in different ontologies. Similarly, relationships are defined between the terms used in different ontologies in COLORE. In this way the relationships represent specific instances of Mapping OPs. Relationships between ontologies themselves are also described so that users may compare their semantics; these relationships are based on the notion of reducibility discussed in the following section.

**Content OPs (CPs)** appear to be the most widely used family of OPs. They are typically domain oriented and provide axioms that are intended to be reused as “building blocks” in order to construct an ontology. CPs can also serve other functions in ontology development such as evaluation. Although they are not necessarily domain-oriented, we view the core theories of COLORE to be examples of useful CPs, as all ontologies in COLORE are reducible to sets of these ontologies. Using the notion of intended models, the core theories in COLORE can also be used for ontology verification ([^35]).

C.1.2 Relationships between Ontologies in COLORE

The sets of ontologies within COLORE are organized based on the notion of the reduction of one ontology to a set of ontologies. This section reviews the background for understanding reduction and the role it plays in organizing ontologies within the repository.

**Hierarchies**

If an ontology is characterized by its set of ontological commitments, then such commitments will be formalized by sets of axioms. Moreover, in order for the commitments to be comparable, their axiomatizations need to be expressed in the same language. Using these intuitions, an ordering can be defined over a set of theories:

**Definition 46.** A hierarchy \( \mathcal{H} = \langle H, < \rangle \) is a partially ordered, finite set of ontologies \( H = T_1, ..., T_n \) such that

1. \( \mathcal{L}(T_i) = \mathcal{L}(T_j) \), for all \( i, j \);
2. \( T_1 \leq T_2 \) iff
   \[ T_1 \models \sigma \Rightarrow T_2 \models \sigma \]
   for any \( \sigma \in \mathcal{L}(T_1) \).

The theories within two hierarchies in COLORE are shown in Figures [C.1] and [C.2]. The Ordering Hierarchy[^1] contains ontologies that axiomatize different classes of orderings, such as partial orderings, linear orderings, trees, and lattices.

The Mereology Hierarchy[^2] contains ontologies that axiomatize different intuitions related to the concept of parthood (see [81] for a full discussion of these ontologies).

Note that all extensions of ontologies in the same hierarchy are nonconservative. An ontology \( T \) is a root ontology iff it is not the extension of any other ontology in the same hierarchy. Within the \( \mathbb{H}^{\text{ordering}} \) Hierarchy, the root ontology is the axiomatization of a transitive relation. Within the \( \mathbb{H}^{\text{mereology}} \) Hierarchy, the root ontology is

[^1]: http://code.google.com/p/colore/source/browse/trunk/ontologies/core/ordering
[^2]: http://code.google.com/p/colore/source/browse/trunk/ontologies/core/mereology
Figure C.1: Ontologies in \( \mathbb{Order} \): the core hierarchy of orderings. Dashed lines denote nonconservative extension. Theories in bold are ones which are used in this work.

Figure C.2: Ontologies in \( \mathbb{Mereology} \): the hierarchy of mereologies. Dashed lines denote nonconservative extension. Theories in bold are ones which are used in this work.
the axiomatization of a basic mereology (in which the parthood relation is transitive, reflexive, and antisymmetric). This ontology is definably equivalent to the theory $T_{\text{partial order}}$ within the $\mathbb{H}_{\text{ordering}}$ Hierarchy.

Reducibility

Definable equivalence is a relationship between two ontologies; this can be generalized to a relationship among sets of ontologies. The basis for this approach is the model-theoretic notion of reducibility introduced in [35].

**Definition 47.** A ontology $T$ is reducible to a set of ontologies $T_1, \ldots, T_n$ iff

1. $T$ faithfully interprets each $T_i$, and
2. $T_1 \cup \ldots \cup T_n$ faithfully interprets $T$.

The set of ontologies $T_1, \ldots, T_n$ in the definition is referred to as the reduction of $T$ in the repository.

It is easy to see that two definably equivalent ontologies are reducible to each other. For example, within COLORE, the ontology $T_{\text{mereology}}$ is reducible to the ontology $T_{\text{linear ordering}}$ and vice versa.

The following result from [36] characterizes the relationship between reducibility and definable equivalence, and it will be used in this work to prove results about reducibility:

**Theorem 13.** Let $T_1, \ldots, T_n$ be a set of ontologies such that $\Sigma(T_i) \cap \Sigma(T_j) = \emptyset$ for all $1 \leq i, j \leq n, i \neq j$.

A ontology $T$ is reducible to $T_1, \ldots, T_n$ iff $T$ is definably equivalent to $T_1 \cup \ldots \cup T_n$.

Later, the reductions of several different ontologies are presented, and their relationship to design patterns is discussed.

Core and Complex Hierarchies

The notion of the reducibility of ontologies can be used to specify an ordering on the set of hierarchies.

**Definition 48.** Let $\mathbb{H}_1, \ldots, \mathbb{H}_n$ be a finite set of hierarchies. A repository $\mathcal{R} = \langle \mathcal{H}, \sqsubseteq \rangle$ is a partially ordered set where

- $\mathcal{H} = \{ \mathbb{H}_1, \ldots, \mathbb{H}_n \}$;
- $\mathbb{H}_i \sqsubseteq \mathbb{H}_j$ iff each root ontology in $\mathbb{H}_j$ has a reduction that contains a ontology $T$ in $\mathbb{H}_i$.

For example, it can be shown that $\mathbb{H}_{\text{ordering}} \sqsubseteq \mathbb{H}_{\text{mereology}}$, since the root ontology in $\mathbb{H}_{\text{mereology}}$ is definably equivalent to the ontology $T_{\text{partial ordering}}$ in $\mathbb{H}_{\text{ordering}}$. On the other hand, $\mathbb{H}_{\text{mereology}} \not\sqsubseteq \mathbb{H}_{\text{ordering}}$, since the root ontology for $\mathbb{H}_{\text{ordering}}$ (which is $T_{\text{transitive}}$) is not reducible to any ontology in $\mathbb{H}_{\text{mereology}}$.

By considering only repositories that contain a finite set of hierarchies, it is guaranteed that the partial ordering $\sqsubseteq$ has minimal elements.

**Definition 49.** A hierarchy $\mathbb{C} = \langle \mathcal{C}, \preceq \rangle$ is a core hierarchy iff it is a minimal hierarchy in the repository $\mathcal{R} = \langle \mathcal{H}, \sqsubseteq \rangle$.

An ontology $T$ is a core ontology theory iff it is in a core hierarchy.

A complex hierarchy $\mathbb{H} = \langle \mathcal{H}, \preceq \rangle$ is a hierarchy which is not minimal in the repository $\langle \mathcal{R}, \sqsubseteq \rangle$.

An ontology $T$ is a complex ontology iff it is in a complex hierarchy.
Figure C.3: Ontologies in $\mathbb{H}_{\text{subposet}}$: the hierarchy of theories of relationships between partially ordered sets. Dashed lines denote nonconservative extension and solid lines denote conservative extension. Ontologies in bold are ones which are used in this work.

Through the notion of reducibility, it is clear that core ontologies play the role of building blocks for all other ontologies within the repository. A complex ontology is either constructed from a set of core ontologies or it is an ontology that imposes additional ontological commitments on a core ontology (e.g. the root theory of the $\mathbb{H}_{\text{mereology}}$ Hierarchy imposes additional ontological commitments that make the parthood relation reflexive and antisymmetric). If the repository contains multiple equivalent core hierarchies, then the reduction will contain multiple definably equivalent core ontologies, and hence there might exist multiple reductions that contain different sets of core ontologies.

Within COLORE, the notion of a core ontology is therefore based on the logical notion of reducibility, rather than on the distinction between generic vs domain ontologies.

C.1.3 Hierarchies as Design Patterns

Core ontologies within the repository can be definably equivalent to multiple ontologies in other hierarchies. In this sense, they play the role of design patterns that are reused to verify other ontologies; that is, they can be used to prove that the intended models of an ontology are isomorphic to the models of the axiomatization of the ontology. This section considers one set of core ontologies in detail and shows its relationship to a surprising variety of other ontologies from remarkably different domains.

Subposet Hierarchy

Each ontology in the Subposet Hierarchy$^3$ is an extension of an ontology from the Mereology Hierarchy and an ontology from the Ordering Hierarchy. The ontologies shown in Figure C.3 form the basis for the $\mathbb{H}_{\text{subposet}}$ Hierarchy. The root ontology $T_{\text{subposet,root}}$ is the union of $T_{\text{m.mereology}}$ and $T_{\text{partial.ordering}}$, and is a conservative extension of each of these ontologies.

---

$^3$http://code.google.com/p/colore/source/browse/trunk/ontologies/core/subposet/
Thus, each model of $T_{\text{subposet,root}}$ (and hence each model of any ontology in the hierarchy) is the amalgamation of a mereology substructure and a partial ordering substructure.

The ontologies shown in Figure C.3 contain additional axioms that constrain how the mereology is related to the partial ordering. In models of $T_{\text{subposet}}$, the mereology is a subordering of the partial ordering. $T_{\text{ideal}}$ strengthens this condition by requiring that the mereology is a subordering of the partial ordering which forms an ideal. In models of $T_{\text{chain,antichain}}$, elements that are ordered by the mereology are not comparable in the partial ordering.

All ontologies within the $\mathbb{H}_{\text{subposet}}$ Hierarchy combine one of the ontologies in Figure C.3 together with one of the ontologies in Figure C.2 and one of the ontologies in Figure C.1. The following sections explore how different ontologies in the $\mathbb{H}_{\text{subposet}}$ Hierarchy serve as design patterns.

**Multimereology Hierarchy**

Motivated by biomedical ontologies such as GALEN and Foundational Model of Anatomy, Bittner and Donnelly (\cite{Bittner2004},\cite{Bittner2005}) have investigated a class of ontologies that combine different kinds of mereological relations. In particular, they axiomatized three relations for part, component, and containment in an ontology which they call $T_{\text{fo,pcc}}$. The subtheory for the part-of relation has models which are isomorphic to a dense mereology with the weak supplementation principle. Models of the subtheory for the component-of relation are isomorphic to discrete mereologies which satisfy the weak supplementation principle as well as what Bittner and Donnelly refer to as the no-partial-overlap property – if $x$ and $y$ are distinct overlapping objects, then either $x$ is a part of $y$ or $y$ is a part of $x$. Finally, models of the subtheory for the contained-in relation are isomorphic to a discrete partial ordering. Models of $T_{\text{fo,pcc}}$ are amalgamations of the models of the three subtheories, and they are referred to as parthood-component-containment structures. Within the COLORE repository, these theories appear in the $\mathbb{H}_{\text{multimereology}}$ Hierarchy.

The ontology for parthood-component-containment structures also contains three axioms that specify how the substructures are combined. The component-of structure is a subordering of the parthood structure, while the relationship between containment and parthood satisfies the following two conditions – parts are contained in the container of the whole and that if a part contains something then so does the whole.

Bittner and Donnelly give an informal description of the models of their ontology, but do not provide a complete characterization of the models up to isomorphism. However, theories within the $\mathbb{H}_{\text{subposet}}$ Hierarchy can be used to verify\textsuperscript{4} that the models of the ontologies are isomorphic to the intended models of Bittner and Donnelly.

**Theorem 14.** $T_{\text{fo,pcc}}$ is definably equivalent to

\[
(T_{\text{tree,mm,mereology}} \cup T_{\text{dense,weak,separative}} \cup T_{\text{subposet}}) \\
\cup(T_{\text{dense,mm,mereology}} \cup T_{\text{discrete,mereology}} \cup T_{\text{lower,preserve}} \cup T_{\text{upper,preserve}}).
\]

In this sense, it is proved that an ontology design pattern is correctly exemplified for given ontology $O$ by proving that the core ontology is definably equivalent to $O$.

The notion of definable equivalence may also be used to extract multiple design patterns from the same ontologies in those cases where an ontology can be decomposed into modules. Recognizing that $T_{\text{fo,pcc}}$ is actually definably equivalent to two different ontologies in the $\mathbb{H}_{\text{subposet}}$ Hierarchy, two ontologies, $T_{\text{ppcmp}}$ and $T_{\text{ppcnt}}$, can be specified which form a modular decomposition of $T_{\text{fo,pcc}}$.

---

\textsuperscript{4}The proofs for all theorems can be found in http://stl.mie.utoronto.ca/colore/subposet-theorems.pdf
Theorem 15. $T_{ppcmp}$ is definably equivalent to the ontology

$$T_{\text{free-mm\_mereology}} \cup T_{\text{dense\_weak\_separative}} \cup T_{\text{subposet}}$$

Theorem 16. $T_{ppcnt}$ is definably equivalent to

$$T_{\text{dense\_weak\_separative}} \cup T_{\text{discrete\_mereology}} \cup T_{\text{lower\_preserve}} \cup T_{\text{upper\_preserve}}$$

It is important to note that $T_{ppcmp}$ and $T_{ppcnt}$ are each definably equivalent to a unique ontology within the $H_{\text{subposet}}$ Hierarchy. As stated in the previous section, each ontology within the $H_{\text{subposet}}$ Hierarchy is a combination an ontology in the $H_{\text{ordering}}$ Hierarchy, an ontology in $H_{\text{mereology}}$ Hierarchy, and one of the “building block” ontologies in Figure C.3 that specifies how the mereology and partial ordering are amalgamated.

Periods Hierarchy

The axioms in the ontologies of the $H_{\text{periods}}$ Hierarchy were first proposed by van Benthem in [84]. The key ontology of this hierarchy, referred to as $T_{\text{period}}$, constitutes the minimal set of conditions that must be met by any period structure and has two relations (precedence and inclusion) and two conservative definitions (for the glb and overlaps relations) as its signature. Transitivity and irreflexivity axioms for the precedence relation make it a strict partial order, and transitivity, reflexivity, and antisymmetry axioms for the inclusion relation make it a partial order; the axioms of monotonicity enforce correct interplay between the precedence and inclusion relations. Van Benthem further includes an axiom that guarantees the existence of greatest lower bounds between overlapping intervals.

Theorem 17. $T_{\text{period}}$ is definably equivalent to the ontology

$$T_{\text{prod\_mereology}} \cup T_{\text{partial\_ordering}} \cup (T_{\text{upper\_preserve}} \cup T_{\text{lower\_reverse}} \cup T_{\text{chain\_antichain}})$$

The relationships between the ontologies in this hierarchy were explored in [66]. In particular, additional theories within the $H_{\text{subposet}}$ Hierarchy were shown to be definably equivalent to various extensions of $T_{\text{period}}$ as axiomatized by van Benthem. This illustrates how we can use design patterns to specify the axiomatization of new ontologies in a hierarchy. Conversely, a subtheory of $T_{\text{period}}$ was used to identify a new ontology within the $H_{\text{subposet}}$ Hierarchy, thus illustrating how we can abstract new design patterns from a set of existing ontologies.

Subactivities in the PSL Ontology

The PSL Ontology uses the $\text{subactivity}$ relation to capture the basic intuitions for the composition of activities. This relation is a discrete partial ordering, in which primitive activities are the minimal elements.

The core ontology $T_{\text{subactivity}}$ alone does not specify any relationship between the occurrence of an activity and occurrences of its subactivities. For example, subactivities $\text{paint}$ and $\text{polish}$ might be composed into some other activity, say $\text{surfacing}$, and $\text{make\_body}$ and $\text{make\_frame}$ may be composed into another activity, say $\text{fabricate}$. However, this specification of subactivities alone does not allow us to say that $\text{surfacing}$ is a nondeterministic activity, or that $\text{fabricate}$ is a deterministic activity.

The primary motivation driving the axiomatization of $T_{\text{atomic}}$ is to capture intuitions about the occurrence of concurrent activities. Since concurrent activities may have preconditions and effects that are not the conjunction of the preconditions and effects of their activities, concurrency in models of $T_{\text{atomic}}$ is represented by the occurrence of one concurrent activity rather than multiple concurrent occurrences.

---

5http://code.google.com/p/colore/source/browse/trunk/ontologies/complex/periods
6http://code.google.com/p/colore/source/browse/trunk/ontologies/complex/psl/subactivity
Atomic activities are either primitive or concurrent (in which case they have proper subactivities). The core ontology\footnote{http://code.google.com/p/colore/source/browse/trunk/ontologies/complex/psl/atomic} $T_{atomic}$ introduces the function $conc$ that maps any two atomic activities to the activity that is their concurrent composition. Essentially, what we call an atomic activity corresponds to some set of primitive activities – every concurrent activity is equivalent to the composition of a set of primitive activities. Although $T_{subactivity}$ can represent arbitrary composition of activities, the composition of atomic activities is restricted to concurrency.

**Theorem 18.** The ontology $T_{subactivity} \cup T_{atomic,act}$ is definably equivalent to the ontology $T_{cem,meremereology} \cup T_{discrete,partialordering} \cup T_{ideal}$

By this Theorem, models of $T_{subactivity} \cup T_{atomic,act}$ are isomorphic to a structure in which a mereological field (on the set of atomic activities) forms an ideal within a discrete partial ordering (on the set of all activities).

**Occurrence Trees in the PSL Ontology**

Within the PSL Ontology, an occurrence tree\footnote{http://code.google.com/p/colore/source/browse/trunk/ontologies/complex/psl/occtree} is a partially ordered set of activity occurrences, such that for a given set of activities, all discrete sequences of their occurrences are branches of the tree. An occurrence tree contains all occurrences of all activities; it is not simply the set of occurrences of a particular (possibly complex) activity. Because the tree is discrete, each activity occurrence in the tree has a unique successor occurrence of each activity.

In addition, there are constraints on which activities can possibly occur in some domain. Although occurrence trees characterize all sequences of activity occurrences, not all of these sequences will intuitively be physically possible within the domain. It is important to consider the subtree of the occurrence tree that consists only of possible sequences of activity occurrences; this subtree is referred to as the legal occurrence tree.

**Theorem 19.** The ontology $T_{pslcore} \cup T_{occtree}$ is definably equivalent to the ontology $T_{tree,meremereology} \cup T_{tree} \cup T_{ideal}$

By this Theorem, models of $T_{pslcore} \cup T_{occtree}$ are isomorphic to a structure in which a tree mereology (on the set of legal activity occurrences) forms an ideal within a tree ordering (on the set of all activity occurrences). It is interesting to notice that $T_{ideal}$ is used both for this ontology as well as for $T_{subactivity}$, demonstrating how one core ontology can be reused as a pattern across very different generic ontologies.

**C.1.4 Summary**

The previous section provided examples of the ways in which core ontologies within COLORE can be utilised as CPs. This illustrates how, with signature translations, a variety of real-world ontologies are in fact comprised of core theories from the same hierarchy, and how even the same core theories were reused with signature translations in different ontologies. Further, it was shown how core ontologies could be used to verify that an ontology contained the desired CPs (core theories), and how new core ontologies (CPs) could be identified by abstracting from ontologies in COLORE.

The ontologies in COLORE's hierarchies (specifically the core theories) correspond well to the definition of CPs provided by \cite{24}: "CPs are small ontologies that mediate between use cases (problem types) and design solutions. They are used as modelling components: ideally, an ontology results from a composition of CPs, with appropriate dependencies between them, plus the necessary design expansion based on specific needs". Based on this definition, each ontology in COLORE could be considered to be a CP as any of the modules could conceivably...
be reused to build other ontologies. However, the focus in this work has been on the core theories as they are most recognizable as CPs - although they are not necessarily domain-oriented, they are definably equivalent to theories that appear in multiple, different domains. They serve as syntactic templates for axioms in a variety of domains, so in a sense they combine aspects of CPs with the more domain-independent Logical OPs.

The relationships defined in COLORE may be considered to be OPs, as the assistance they provide for ontology development is similar to the aid provided by OPs. Nevertheless, some of the features of COLORE offer capabilities beyond what is currently offered by the OP community. For example, because of the formalized nature of the relationships specified in COLORE, automated reasoning can be implemented to verify the mappings between the ontologies. In addition, the notion of reducibility can be implemented to identify useful CPs from ontologies in COLORE – the more theories that are reducible to a particular core theory, the more useful it is. Automated theorem provers may also be used to verify that an ontology is in fact a core theory. Lastly, OPs were not intended to be restricted to a particular representation language [24] and the use of first-order logic in COLORE supports this ideal as patterns from a wide range of languages may be represented.

C.2 Logical Language Translation: Using PSL to Extend Event Ontologies

The notion of events is pervasive, so it is natural to find that much existing work on the Semantic Web in some way addresses the challenge of representing and integrating event-related data. While integration is certainly a valuable application, more complex reasoning can and should be performed with the event information on the web. Given the strides that have been made for integration and basic information retrieval for event-related information, it is logical to now ask what can be done with this information. Many existing event ontologies tend to take a rather simplistic approach in order to better facilitate use with many diverse sources, resulting in a rather limited semantics. It is therefore unclear to what extent these theories can support the non-trivial reasoning problems that are required for applications such as complex event processing. The aim of this work is to address the following questions: What reasoning problems can be done with the status quo? What gains can we make by augmenting their axiomatizations? What kinds of queries require more substantial changes to the ontologies?

To investigate this, a set of competency questions (CQs) is identified in Section C.2.2, motivated by several potential application domains related to complex event processing; these questions are representative of some of the more complex reasoning tasks of interest. Three well-known event ontologies are then extended via the technique of grafting them with a logical language translation of the Process Specification Language (PSL) ontology in Section C.2.3. The resulting set of event ontologies is then evaluated against the competency questions.

The outcomes of the evaluation, presented in Section C.2.4 demonstrate many attainable opportunities to achieve substantial gains in the reasoning abilities of these Semantic Web ontologies. The results and their implications on the possibilities for increased reasoning on the Semantic Web are considered in Section C.2.4. This material first appeared in [52].

C.2.1 Background

As mentioned in the previous section, there currently exist numerous efforts toward the development of ontologies for the representation of events on the Semantic Web. Although perhaps contrary to some of the underlying
principles of ontologies (e.g. reuse), this is not entirely surprising given that the concept of an event plays an important role in such a wide variety of contexts. A review of existing event ontologies agrees with this observation as we have found the application areas for existing work to be quite diverse, including domains such as artefacts, historic events, and business processes. The following sections present a more detailed review of three of the more prominent generic event ontologies used for this investigation.

SEM

The Simple Event Model (SEM) ontology [85] was designed to represent event information on the web. Formalized in RDF, it stresses minimal semantic commitments, an approach that is intended to facilitate “maximal interoperability” with the variety of event information on the web. SEM contains four “core” classes: Event, Actor, Place, and Time, aimed at describing what occurred, who (or what) participated in the occurrence, where it occurred, and when.

Each of these core classes has an associated Type class (i.e. EventType, ActorType, etc) to allow for the specification of distinctions between the core classes. A Constraint class is also defined (with subclasses View, Temporary, and Role) to allow for some description of the properties of an event (e.g. to describe the nature of an actor’s participation in an event).

Although there is some discussion of reasoning with the use of the SEM Prolog API, the demonstrated functionality is essentially limited to look-up queries; the application focus appears to be on event information integration.

LODE

The ontology for Linking Open Descriptions of Events (LODE) [77] also approaches the information integration problem – but in this case with the broader goal of creating an ontology that can also serve as an interlingua for existing event ontologies, thereby also (potentially) integrating their information. The view of the event domain taken here is focused more on the subject matter of journalists and historians – what might loosely be called world events.

Axiomatized in OWL, LODE aims to achieve this interoperability through a set of what they refer to as mapping axioms, (it should be noted that the definition and use of such axioms differs from the perspective taken in this work) between its concepts, and those of some existing event ontologies. In an effort to ensure interoperability, LODE focuses on representing what they refer to as the “factual” aspects of an event, core concepts which they believe are not subject to interpretation. LODE provides an Event class along with the properties: atPlace, atTime, circa, illustrate, inSpace, involved, and involvedAgent to describe an event.

The Event Ontology

The Event Ontology [10] evolved out of the Music Ontology project [25], which was developed to integrate music-related information from heterogeneous sources. Perhaps most notably, the Event Ontology is referenced in the development of the BBC Sports Ontology [11] which was implemented to facilitate automated curation of the BBC’s world cup site.

Axiomatized in RDFS, the Event Ontology consists of the classes: Event, Factor, and Product; and the properties: agent, factor, literal_factor, place, product, sub_event, and time. Similar to LODE, (but without the explicit
aim of facilitating interoperability) the Event Ontology also includes concepts from other ontologies, such as foaf[12] and the WGS84 Geo Positioning Ontology[13]

PSL

With its rich, rigorous axiomatization of such concepts, PSL was a natural choice for the task of grafting the chosen Semantic Web ontologies onto a more expressive ontology. PSL is an ontology designed to facilitate the correct and complete exchange of process information among manufacturing systems[32]. These applications include scheduling, process modelling, process planning, production planning, simulation, and project management. The PSL ontology $T_{psl}$[14] is organized into PSL-Core and a set of partially ordered extensions; the core ontology consists of four disjoint classes: activities can have zero or more occurrences, activity occurrences begin and end at time points, time points constitute a linear ordered set with end points at infinity, and objects are elements that are not activities, occurrences, or time points[32]. There are five additional modules within the PSL Ontology – $T_{acttree}$ (which is closely related to situation calculus), $T_{subactivity}$ (which axiomatizes the composition relation on activities), $T_{atomic}$ (which axiomatizes concurrent activities), $T_{complex}$ (which axiomatizes complex activities), and $T_{actocc}$ (which axiomatizes the composition relation on occurrences of complex activities).

More recently, the PSL Ontology has been extended to capture the relations between activity occurrences, actors, locations, and time intervals. In particular, $T_{psl\_location}$[15] merges $T_{psl}$ with a multidimensional mereotopology that represents containment relations among spatial entities. The ontology $T_{psl\_actor}$[16] specializes the participates_in relation from PSL-Core by introducing the relation performed_in between actors and the activity occurrences that they perform.

C.2.2 Generic Requirements

Up to this point, the opportunities of reasoning about events have been considered only at a high level. The aim of this section is to further motivate this work by identifying practical application domains where reasoning about events would offer significant value. The scenarios described below are quite diverse, yet the core of their reasoning problems is relatively general. Based on the commonly adopted scope of events as including the notion of participants and locations, a set of competency questions capable of providing useful support in the variety of the motivating scenarios described below is then elicited.

Motivating Scenarios

Potential reasoning problems with applications for emergency response centres, city services management, and context awareness may readily be identified. Some of these scenarios are inspired by the application domains described by the various existing event ontologies. The following scenarios illustrate the different sorts of problems where reasoning about events could be valuable. Essentially, they stem from the question – assuming that the existing ontologies have successfully integrated event information in their various application domains, what can we do with it? Although these scenarios are among many potential applications that include the notion of events, it is reasonable to speculate that certain patterns and types of questions are likely to arise repeatedly in many other,
unrelated applications. The scenarios, described below, were used to motivate the set of general, event-oriented competency questions which are presented in Section C.2.2.

**Emergency Response Centre** When responding to an incident report, any additional information could be valuable to prepare the dispatched units en route. Integrated event information could be leveraged to provide this additional information to the dispatch centre, by allowing the dispatchers to pose queries in order to identify information that might be relevant to a particular incident.

**City Services Management** Knowledge of planned events as well as the ability to reason and analyse past events could serve to better (and more easily) inform city workers in the planning of various projects, as well as to assist management in identifying potential issues or trends related to various aspects of city maintenance.

**Context-awareness** Consider the variety of opportunities for context-aware applications for personal use. Information about what events are occurring, or can occur in a city could be leveraged to better inform an individual how best to navigate to their destination, or complete a set of errands. Similar tools could also be applied to aid users in navigating or planning for recreational events (festivals, etc).

**Informal Competency Questions**

The CQs are first presented informally here, in natural language; followed by an overview of the test domain theory, necessary for the ontologies’ evaluation with automated reasoners. For the subsequent evaluation, the competency questions are initially formalized in the vocabulary of PSL, since it has the broadest scope of the event ontologies to be evaluated. Further, since each of the Semantic Web event ontologies will be extended by PSL as part of this investigation, the translation of each of the CQs, (as well as the domain theory) will be straightforward from the identified mappings.

The following set of generic CQs are essentially patterns that are applicable in any of the motivating scenarios described previously:

**CQ1** What actors may have participated in some activity occurrence?

**CQ2** What can possibly occur next after an occurrence of some activity?

**CQ3** Are occurrences of two activities possibly subactivity occurrences of the same complex activity occurrence?

**CQ4** Are any other activities possibly taking place at the same place and the same time as a particular activity?

**CQ5** Assuming that occurrences of two activities are part of the same overall activity occurrence, what activities possibly occurred between them?

**CQ6** What activity could have occurred before an occurrence of some other observed activity?

**CQ7** Is there an activity that will definitely occur after an occurrence of some activity?

**CQ8** What activities are scheduled to occur at a given time and location?

**CQ9** During what time intervals are no events occurring at a given location?

**CQ10** Do any occurrences of two activities overlap?

---

For the complete set of formalized CQs, the reader is referred to: [http://stl.mie.utoronto.ca/ontologies/CQs.txt](http://stl.mie.utoronto.ca/ontologies/CQs.txt)

These questions are expressed generically, but the reader should notice that they may easily be specialized to the motivating scenarios described previously. For example, CQ1 might become “What contractors may have performed road repairs?”
CQ11 What subactivities of some activity is a particular actor participating in?
CQ12 Is an actor of interest possibly participating in an activity?
CQ13 Given observed occurrences of two activities, what might an actor of interest participate in next?

C.2.3 Extensions of the Event Ontologies

One of the primary questions addressed here is whether existing event ontologies on the Web can better support reasoning about events (as specified by the competency questions) if they are extended by additional axioms. A key step in this endeavor is to determine the relationships between the PSL Ontology and the Semantic Web event ontologies. This enables the specification of extensions of the event ontologies; further, it provides the opportunity for a model-theoretic evaluation of these ontologies, as well as the identification of any relationships among the different event ontologies themselves. In this section, axiomatization of subtheories of PSL that are definable in the description logic SROIQ and in the Semantic Web Rule Language (SWRL), (which is equivalent to the extension of OWL with the Horn sublanguage of FOL) are discussed. The notion of ontology grafting is then introduced and used to specify maximal extensions of the event ontologies with respect to the SROIQ and SWRL axiomatizations of PSL. In the next section, these extensions will be evaluated with respect to the competency questions presented earlier.

Two challenges arise – the relationship between the axiomatizations of different ontologies in the same logic, as well as the relationship between the axiomatizations of a given ontology in different logics. To address this, the following notion of a logic mapping is introduced:

**Definition 50.** Let $T_1$ be a theory in a logic $L_1$ and let $T_2$ be a theory in a logic $L_2$.

$T_1$ is language-equivalent to $T_2$ iff $T_1$ is logically equivalent to the translation of $T_2$ under the logic mapping from $L_1$ to $L_2$.

We will use the logic mapping from SROIQ to FOL specified in [57] when specifying the language-equivalence of SROIQ and FOL theories.

**Ontology Grafting**

The basic relationship between theories $T_A$ and $T_B$ is the notion of interpretation [37], which is a mapping from the language of $T_A$ to the language of $T_B$ that preserves the theorems of $T_A$. The interpretation is faithful if the mapping also preserves the satisfiable sentences of $T_A$. If there is an interpretation of $T_A$ in $T_B$, then there exists a set of sentences (referred to as translation definitions) in the language $L_A \cup L_B$ of the form

$$(\forall \bar{x}) \ p_i(\bar{x}) \equiv \varphi(\bar{x})$$

where $p_i(\bar{x})$ is a relation symbol in $L_A$ and $\varphi(\bar{x})$ is a formula in $L_B$.

When applied with the Semantic Web event ontologies, there is an additional problem in that the translation definitions used to specify interpretations among theories are not definable either within SROIQ or SWRL. The approach of ontology mapping needs to be modified so that the notion of faithful interpretation to first-order theories can be used, and then the resulting theories may be translated into SROIQ and SWRL. The notion of ontology grafting is introduced to accomplish this, in which one ontology is extended via the translation definitions specified using the first-order translations of a set of other ontologies.
Definition 51. An ontology $T_3$ is the grafting of the ontology $T_2$ onto the ontology $T_1$ iff there exists $T_1', T_2', T_3'$ such that

1. $T_3'$ is a nonconservative extension of $T_2'$ such that both theories have the same signature;
2. $T_1'$ faithfully interprets $T_3'$;
3. $T_i$ is language-equivalent to $T_i'$.

In the rest of this section, the event ontologies $T_{\text{sem}}$, $T_{\text{event}}$, and $T_{\text{lode}}$ are grafted onto $T_{\text{psl}}$. In doing so, maximal extensions of the event ontologies which are language-equivalent to subtheories of the PSL Ontology are specified. In particular, $T_{\text{psl}}_{\text{dl}}$ which is language-equivalent to $T_{\text{sroiq}}^{\text{psl}}$ and $T_{\text{psl}}_{\text{swrl}}$ which is language-equivalent to $T_{\text{swrl}}^{\text{psl}}$ will be used.

**OWL Extensions of the Ontologies**

**SEM**

Let $T_{\text{sem}}$ be the first-order theory which is language-equivalent to the subtheory of the SROIQ theory $T_{\text{sroiq}}^\text{psl}$ that omits the axioms for the timestamp datatype properties.

Lemma 5. $T_{\text{sem}}$ is interpreted by $T_{\text{psl}}_{\text{dl}}$.

Proof. Let $\Delta_1$ be the following set of translation definitions:

\[
(\forall x) \text{sem:EventType}(x) \equiv \text{activity}(x)
\]

\[
(\forall x) \text{sem:Event}(x) \equiv \text{activity}\_\text{occurrence}(x)
\]
(∀x) sem:Time(x) ≡ timepoint(x) ∨ timeinterval(x)

(∀x, y) sem:eventType(x,y) ≡ occurrence(x,y)

(∀x, y) sem:hasSubEvent(x,y) ≡ subactivity(x,y)

(∀x, y) sem:hastime(x,y) ≡ (psl_interval(x,y) ∨ begins(x,y) ∨ ends(x,y))

(∀x) sem:Actor(x) ≡ actor(x)

(∀x) sem:Place(x) ≡ location(x)

(∀x, y) sem:hasPlace(x,y) ≡ occurred(x,y)

(∀x, y) sem:hasActor(x,y) ≡ performed(y,x)

We can use Prover9 to show that $T_{psl,dl} \cup \Delta_1 \models T_{sem}$.

It is important to realize that this interpretation is not faithful because there are sentences in the signature of $T_{sem}$ which are entailed by the interpretation but which are not entailed by $T_{sem}$ itself. The key idea is that the set of such sentences can be used to extend $T_{sem}$ until a theory which is faithfully interpreted by $T_{psl,dl}$ is found. This extension is referred to as $T_{sem,x}$ and the translation of the resulting theory to be $T_{sroiq,sem}$. Since $T_{psl,dl} \cup \Delta_1$ is a conservative extension of $T_{sem,x}$, $T_{sem,x}$ is faithfully interpreted by $T_{psl,dl}$.

**Theorem 20.** $T_{sroiq,sem}$ is the grafting of $T_{sroiq,sem}$ onto $T_{psl,dl}$.

Thus, $T_{sroiq,sem}$ is a maximal extension of $T_{sroiq,sem}$ which has the same signature as $T_{sroiq,sem}$ any stronger extension would require an expanded signature.

A summary of the results is shown in Figure C.4

**Event Ontology**

Let $T_{event}$ be the first-order theory which is language-equivalent to the SROIQ theory $T_{sroiq,event}$.

**Lemma 6.** $T_{event}$ is interpreted by $T_{psl,dl}$.

$T_{event}$ can be extended with the additional sentences which are entailed by $T_{psl,dl} \cup \Delta_2$; this nonconservative extension is referred to as $T_{event,x}$. Each of these sentences can also be axiomatized in SROIQ, and the resulting theory is referred to as $T_{sroiq,event,x}$. As with the Simple Event Model, the ontology $T_{event,x}$ is a maximal extension of $T_{event}$ within its hierarchy.

**Theorem 21.** $T_{sroiq,event,x}$ is the grafting of $T_{sroiq,event}$ onto $T_{sroiq}$.

In terms of the definition of ontology grafting, it is clear that $T_{event,x}$ is a nonconservative extension of $T_{event}$ and that $T_{psl,dl}$ faithfully interprets $T_{event,x}$.

---

26 https://www.cs.unm.edu/~mccune/mace4/
27 http://colore.oor.net/simple_event_model/sem_x.clif
28 stl.mie.utoronto.ca/ontologies/simple_event_model/sem_x.owl
29 http://colore.oor.net/event_ontology/event.clif
30 stl.mie.utoronto.ca/ontologies/event_ontology/event.owl
31 http://colore.oor.net/event_ontology/event_x.clif
32 stl.mie.utoronto.ca/ontologies/event_ontology/event_x.owl
LODE

Let $T_{	ext{lode}}$ be the first-order theory which is language-equivalent to $T_{	ext{lode x}}$.

Lemma 7. $T_{	ext{lode}}$ is interpreted by $T_{	ext{psl dt}}$.

As with the other two event ontologies, $T_{	ext{lode}}$ may be extended with the additional sentences which are entailed by $T_{	ext{psl dt}} \cup \Delta_3$, resulting in $T_{	ext{event x}}$. Each of these sentences can also be axiomatized in SROIQ, and the resulting ontology is $T_{	ext{lode x}}$. As with the other event ontologies, the extension of $T_{	ext{lode}}$ is conservative, so that $T_{	ext{lode x}}$ is faithfully interpreted by $T_{	ext{psl dt}}$. Therefore,

Theorem 22. $T_{	ext{sroiq lode x}}$ is the grafting of $T_{	ext{lode}}$ onto $T_{	ext{sroiq psl dt}}$.

SWRL Extensions of the Ontologies

Looking at the axiomatizations of the event ontologies, it is clear that the restriction to OWL omits axioms that may be required to support the competency questions. This section considers extensions of the event ontologies in which the additional expressiveness of SWRL is exploited.

Theorem 23. $T_{	ext{swrl sem r}}$ is the grafting of $T_{	ext{sroiq sem r}}$ onto $T_{	ext{swrl psl dt}}$.

Proof. Let $T_{	ext{sem r,swrl}}$ be the first-order ontology which is language-equivalent to $T_{	ext{swrl sem r}}$. Using the translation definitions from the proof of Theorem 5, we have $T_{	ext{psl,swrl}} \cup \Delta_1 \models T_{	ext{sem r,swrl}}$.

Theorem 24. $T_{	ext{swrl event r}}$ is the grafting of $T_{	ext{sroiq event r}}$ onto $T_{	ext{swrl psl dt}}$.

Proof. Let $T_{	ext{event r,swrl}}$ be the first-order ontology which is language-equivalent to $T_{	ext{swrl event r}}$. Using the same translation definitions as in the proof of Lemma 6, we have $T_{	ext{psl,swrl}} \cup \Delta_2 \models T_{	ext{event r,swrl}}$.

Theorem 25. $T_{	ext{swrl lode r}}$ is the grafting of $T_{	ext{sroiq lode r}}$ onto $T_{	ext{swrl psl dt}}$.

Proof. Let $T_{	ext{lode r,swrl}}$ be the first-order ontology which is language-equivalent to $T_{	ext{swrl lode r}}$. Using the same translation definitions as in the proof of Lemma 7, we have $T_{	ext{psl,swrl}} \cup \Delta_3 \models T_{	ext{lode r,swrl}}$.
C.2.4 Evaluation

In order to demonstrate the evaluation of these competency questions, a domain theory must be specified, i.e. a set of individuals of the classes of the ontology. Note that the evaluation of these competency questions is meant to demonstrate the scope (both in terms of lexicon and semantics) distinctions between the event ontologies; given that the size and complexity of the domain theory has no impact on the scope of the ontology, a toy scenario is sufficient for this purpose. The generic domain theory employed consists of two complex activities, \( A_1, A_2 \), as well as five atomic activities \( (A_{21}, A_{22}, A_3, A_4, A_5) \), each with varying possible occurrences and orderings. Additional information is also specified regarding times and locations of occurrences and participating actors, as well as possible locations and participation for activities. The complete domain theory specifications for each theory to be evaluated may be found either embedded in the OWL ontologies, or in the related input files (in the case of first-order logic proofs).

In transitioning from the informal set of competency questions to a formal specification of queries, there are often a variety of subtle distinctions in the way the queries could be interpreted. For example, CQ1, when made more specific, can be interpreted in several ways:

**CQ1-1** What actors participated in the occurrence, \( O_{21} \)?

**CQ1-2** What actors perform \( A_2 \)?

**CQ1-3** What actors participated in some occurrences of \( A_2 \)?

Each such interpretation may result in a distinct query when formalized, which may impact the ontologies’ ability to represent or answer it. For example, notice that the first interpretation (CQ1-1) is expressible by all ontologies, whereas none of the Semantic Web event ontologies are able to express the second interpretation (CQ1-2). All such recognized alternate interpretations have been included in the evaluation to avoid excluding any potentially interesting results.

**CQ1-1**:
- first-order logic: \((\exists a)(actor(a) \land performed\_in(a, O_{21}))\)
- psl.owl: Actor and performed\_in value \( O_{21} \)
- The Event Ontology: Agent and agent value \( O_{21} \)
- SEM: Actor and inverse (‘has Actor’) value \( O_{21} \)
- LODE: Agent and ‘involved agent’ value \( O_{21} \)

**CQ1-2**:
- first-order logic: \((\exists a)(actor(a) \land performs(a, A_2))\)
- psl.owl: Actor and performs value \( A_2 \)
- Query out of the scope of SEM, LODE, and the Event Ontology’s lexicon.

**Results**

Using the HermiT 1.3.8 plug-in provided by Protege version 4.3 [http://protege.stanford.edu/](http://protege.stanford.edu/) each of the original event ontologies, their extensions via ontology grafting onto \( T_{psl}^{sroiq} \) and \( T_{psl}^{swrl} \), as well as \( T_{psl}^{sroiq} \) and \( T_{psl}^{swrl} \) were evaluated against the formalized competency questions.
In the case of certain queries it was found that a formalization could not be specified in OWL. Here, the first-order logic translations of each of the OWL theories (available online, as referenced in the previous section) were utilized, and an evaluation using the first-order automated theorem prover, Prover9, was attempted. The idea behind this was that a positive result would demonstrate that the axioms specified in the ontology were in fact sufficient to answer the query; this avoids faulting the ontology for an issue of query language expressivity/tool support. Only in the case that a query was still not provable in the first-order translation would it be inferred that the axioms were too weak.

The results of each ontology against the reasoning problems are summarized in Table C.2.4.

<table>
<thead>
<tr>
<th>CQ</th>
<th>Entailed By</th>
<th>Expressible By</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>psl.owl, psl_dl.clif, psl.swrl, psl_swrl.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif, event.x.owl, event.x.clif, event.r.swrl, event.r.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif, sem.owl, sem.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif, event.owl, event.clif, event.x.owl, event.x.clif, event.r.swrl, event.r.clif, lode.owl, lode.clif, lode.x.owl, lode.x.clif, lode.r.swrl, lode.r.clif</td>
</tr>
<tr>
<td>1-2</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>1-3</td>
<td>psl.owl, psl_swrl.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif, sem.owl, sem.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>psol_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>3-1</td>
<td>psl.owl, psl_swrl.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif, sem.owl, sem.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif</td>
</tr>
<tr>
<td>3-2</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>6</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>7</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif, sem.owl, sem.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif, event.owl, event.clif, event.x.owl, event.x.clif, event.r.swrl, event.r.clif, lode.owl, lode.clif, lode.x.owl, lode.x.clif, lode.r.swrl, lode.r.clif</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>11</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>12-1</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
<tr>
<td>12-2</td>
<td>psl.owl, psl_swrl.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif, sem.owl, sem.clif, sem.x.owl, sem.x.clif, sem.r.swrl, sem.r.clif</td>
</tr>
<tr>
<td>13</td>
<td>psl.owl, psl_swrl.clif</td>
<td>psl.owl, psl_dl.clif, psl_swrl.clif</td>
</tr>
</tbody>
</table>

Table C.1: A high-level summary of the evaluation results
A detailed summary of the evaluation results is available online\textsuperscript{44} Note that although all ontologies were capable of formalizing CQ9, none were able to return a solution as the test domain theory did not include the use of closure axioms.

**In OWL** Both the Event Ontology and LODE had major scope limitations which prevented all but a single competency question from being expressible in their lexicon (CQ1-1). Further, neither ontology was able to return the answer to this question as both lacked sufficient semantics to make the necessary inferences. SEM fared comparably better as its lexicon was able to formalize a total of four of the CQs (CQ1-1, CQ1-3, CQ3-1, and CQ12-2). However, similar to its counterparts, SEM also lacked the necessary axioms to answer any of the four CQs.

Not surprisingly, given its broad scope, the signature of \( T_{psl}^{sroiq} \) supported the specification of all but three of the CQs. In fact, the three CQs that were not expressible (CQ2, CQ5, and CQ10) were hindered not by the scope of the lexicon, but by the limitations of the OWL query language. In terms of reasoning abilities, \( T_{psl}^{sroiq} \) also fared better; the correct answer was returned for all of the expressible CQs, apart from CQ4 and CQ8.

When grafted to \( T_{psl}^{sroiq} \), improvements were achieved in each of the resulting event ontology extensions, with the exception of \( T_{lode,x}^{psl} \); it was still insufficient to answer the query. It was found that the Event Ontology extension was sufficient to return the answer to CQ1-1. Further, \( T_{sem}^{sroiq} \) was capable of correctly solving all four of the CQs that were in its scope.

**In First-Order Logic** The queries CQ2, CQ5, and CQ10 were outside of the expressive capabilities of the DL query tool, and therefore attempted in Prover9\textsuperscript{45}. No evaluation of the queries for LODE, the Event Ontology, or SEM was possible (likewise with their extensions) as the necessary concepts were outside of the ontologies’ lexicons. Using a translation of \( T_{psl}^{sroiq} \) (i.e., \( T_{psl,dl}^{sroiq} \), a proof was obtained, yielding the correct answer to CQ4\textsuperscript{46}. Neither a proof nor a counterexample was found by Prover9 for both CQ5 and CQ10, although a manual counterexample can be found in both cases, showing that the subtheory of the PSL Ontology which is definable in SROIQ is not strong enough to entail solutions to the competency questions.

Since SWRL is an extension of OWL, any competency question entailed by an OWL axiomatization is also entailed by the SWRL axiomatization. In evaluating the adequacy of the SWRL extensions of the Semantic Web event ontologies, we therefore only needed to consider lode\_r\_swrl against CQ1-1, as all of the other competency questions not entailed by the Semantic Web event ontologies were not expressible. The extension lode\_r\_swrl was still unable to infer the correct answer to CQ1-1; its scope is too restricted to allow for the necessary extension (specifically, the subactivity_occurrence relation).

**Troubleshooting Ontology Expressivity**

The results of the evaluation are generally quite encouraging for the goal of supporting more complex reasoning about events on the Semantic Web. \( T_{psl}^{sroiq} \) served as a particularly motivating example of the potential functionality that can be achieved. Further, and specifically with respect to the gains illustrated with \( T_{sem,x}^{sroiq} \), the results indicate the effectiveness of ontology grafting. It should also be noted that the theories of actors and locations translated to create \( T_{psl}^{sroiq} \) were designed to be root theories in COLORE \textsuperscript{37}. In other words they are intentionally weak and make minimal commitments; therefore there likely is further potential for increased reasoning abilities with the creation of even stronger theories.

\textsuperscript{44}http://stl.mie.utoronto.ca/ontologies/results_summary_reformat.pdf
\textsuperscript{45}http://stl.mie.utoronto.ca/ontologies/psl_dl.owl.in
\textsuperscript{46}http://stl.mie.utoronto.ca/ontologies/psl_cq2.proof
Where reasoning is limited, the cause is lack of expressivity in one or multiple ways: insufficient scope of concepts (lexicon), insufficient axioms (semantics), or limitations of the language itself (logic). The results indicate that a key factor in the reasoning limitations of existing ontologies is the scope of their lexicons. This observation could be a cause of the lack of reasoning about events on the Semantic Web, but it is more likely a symptom of the lack of focus on such applications. This may be due to the fact that the integration and search-oriented applications represent more low-hanging fruit for ontology developers, or it could be that the widely adopted approach of using lightweight axiomatizations for integration is simply not conducive to creating ontologies capable of supporting more sophisticated reasoning. While it is logical for the task of information integration to be tackled prior to reasoning tasks, to fully benefit from these previous efforts, the focus should now shift to potential reasoning applications.

In any case, analysis of the results highlights two commonly omitted concepts contribute to the event ontologies’ inability to solve the reasoning problems: the activity/occurrence distinction, and the notion of an ordering over occurrences. It was SEM’s inclusion of the notion of an activity (“event type”) that provided the ability to represent multiple CQs, over and above its peers. Although this may seem like a PSL-specific distinction, it is one which is necessary to ask more interesting questions about events. When limited to occurrences, reasoning becomes restricted to simply asking about particular instances. Without the activity/occurrence distinction the ability to pose queries regarding occurrences of a particular kind is lost. Returning to the motivating scenarios presented initially, this means that queries such as: is there some sort of construction event occurring at a location? are not possible.

None of the selected event ontologies provided a definition for ordering over occurrences. Instead, only some notion of time is associated with an event – likely intended to be instantiated with, or otherwise attached to some form of a timestamp. This over-reliance on datatype-oriented representations (in conjunction with the lack of an activity/occurrence distinction) precludes the reasoning about potential, future orderings of possible events. By relying on timestamps associated with particular occurrences, questions such as – is it possible that flooding occurred prior to the reported power outage? cannot be asked.

C.2.5 Summary

Existing work with event ontologies has focused heavily on the task of representing event information for integration, leaving the task of automated reasoning relatively untouched. We motivated our goal of increased reasoning about events on the Semantic Web, and used this motivation as a source of pragmatic reasoning problems for our evaluation. Through our investigation, we have demonstrated some of the reasoning abilities that Semantic Web ontologies are capable of supporting. Further, we analysed our findings in order to offer explanations for the current lack of such reasoning.

The notion of ontology grafting may serve as the basis for the reuse of first-order logic ontologies to augment theories in less expressive languages. Although these ontologies often have relatively weak axiomatizations, other ontologies such as PSL have rich axiomatizations, albeit sometimes in a different language. Our approach outlines the technique of grafting more expressive ontologies to less-expressive, integration-oriented ontologies as a means of augmenting their axiomatizations and consequently their reasoning power. This approach appears to be particularly well-suited to the reuse of existing ontologies via logical language translation. The formal nature of this approach also lays the groundwork for interoperability between the ontologies being extended. With the mappings created, we could potentially use the signature of PSL to query information represented by any of the three ontologies. There is also the potential to perform model-theoretic verification of these ontologies, as in [37]. We hope that this work may serve not only to motivate continued efforts towards reasoning on the Semantic Web,
but also as a guide for those looking to reuse and reason with existing ontologies.